

DOCUMENT RESUME

ED 198 375

CE 028 094

TITLE Military Curricula for Vocational & Technical Education. Programmer/Analyst 4-4.

INSTITUTION Department of the Army, Washington, D.C.: Ohio State Univ., Columbus. National Center for Research in Vocational Education.

SPONS AGENCY Bureau of Occupational and Adult Education (DHEW/OE), Washington, D.C.

PUB DATE Sep 78

NOTE 711p.; Not available in paper copy due to small and broken print.

EDRS PRICE MF04 Plus Postage. PC Not Available from EDRS.

DESCRIPTORS Behavioral Objectives; *Computer Programs; Computers; *Computer Science Education; Course Descriptions; Curriculum Guides; Learning Activities; Postsecondary Education; Programed Instructional Materials; Programers; *Programing; *Programing Languages; Secondary Education; Technical Education

IDENTIFIERS *COBOL Programming Language; Military Curriculum Project

ABSTRACT

This program of instruction and various instructional materials for a secondary-postsecondary level course for programmer/analysts is one of a number of military-developed curriculum packages selected for adaptation to vocational instruction and curriculum development in a civilian setting. The eight-week, three-section course is designed to provide the skill to program electronic computers structured in COBOL and to code job streams using IBM, DOS, JCL, and utility programs. The program of instruction suggests a time schedule and gives the learning objective and reference(s) for each topic. Section 1, Data Representation, consists of a programmed text covering the binary, octal, and hexadecimal systems. Section 2, Basic COBOL Programming, contains a text and problem exercises with some answers. Topic areas include processing and updating a sequential file, producing an edited report and a report with calculations, processing external and internal tables, and debugging syntax errors. Section 3, Operating Systems, includes a text and programmed text. It focuses on disk operating systems (DOS) organization and operation. Specific topics include coding DOS Job Streams, DOS Librarian Programs, and DOS Utilities and Sorts.

(YLB)

* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *

This military technical training course has been selected and adapted by The Center for Vocational Education for "Trial Implementation of a Model System to Provide Military Curriculum Materials for Use in Vocational and Technical Education," a project sponsored by the Bureau of Occupational and Adult Education, U.S. Department of Health, Education, and Welfare.

MILITARY CURRICULUM MATERIALS

The military-developed curriculum materials in this course package were selected by the National Center for Research in Vocational Education Military Curriculum Project for dissemination to the six regional Curriculum Coordination Centers and other instructional materials agencies. The purpose of disseminating these courses was to make curriculum materials developed by the military more accessible to vocational educators in the civilian setting.

The course materials were acquired, evaluated by project staff and practitioners in the field, and prepared for dissemination. Materials which were specific to the military were deleted, copyrighted materials were either omitted or approval for their use was obtained. These course packages contain curriculum resource materials which can be adapted to support vocational instruction and curriculum development.

The National Center Mission Statement

The National Center for Research in Vocational Education's mission is to increase the ability of diverse agencies, institutions, and organizations to solve educational problems relating to individual career planning, preparation, and progression. The National Center fulfills its mission by:

- Generating knowledge through research
- Developing educational programs and products
- Evaluating individual program needs and outcomes
- Installing educational programs and products
- Operating information systems and services
- Conducting leadership development and training programs

FOR FURTHER INFORMATION ABOUT Military Curriculum Materials

WRITE OR CALL

Program Information Office
The National Center for Research in Vocational
Education
The Ohio State University
1960 Kenny Road, Columbus, Ohio 43210
Telephone: 614/486-3655 or Toll Free 800/
848-4815 within the continental U.S.
(except Ohio)


THE NATIONAL CENTER
FOR RESEARCH IN VOCATIONAL EDUCATION
The Ohio State University - 1960 Kenny Road, Columbus, Ohio 43210
Tel: (614) 486-3655
Cable: CTVCEBOSU/Columbus, Ohio

Military Curriculum Materials for Vocational and Technical Education

Information and Field
Services Division

The National Center for Research
in Vocational Education



Military Curriculum Materials Dissemination Is . . .

an activity to increase the accessibility of military-developed curriculum materials to vocational and technical educators.

This project, funded by the U.S. Office of Education, includes the identification and acquisition of curriculum materials in print form from the Coast Guard, Air Force, Army, Marine Corps and Navy.

Access to military curriculum materials is provided through a "Joint Memorandum of Understanding" between the U.S. Office of Education and the Department of Defense.

The acquired materials are reviewed by staff and subject matter specialists, and courses deemed applicable to vocational and technical education are selected for dissemination.

The National Center for Research in Vocational Education is the U.S. Office of Education's designated representative to acquire the materials and conduct the project activities.

Project Staff:

Wesley E. Budke, Ph.D., Director
National Center Clearinghouse
Shirley A. Chase, Ph.D.
Project Director

What Materials Are Available?

One hundred twenty courses on microfiche (thirteen in paper form) and descriptions of each have been provided to the vocational Curriculum Coordination Centers and other instructional materials agencies for dissemination.

Course materials include programmed instruction, curriculum outlines, instructor guides, student workbooks and technical manuals.

The 120 courses represent the following sixteen vocational subject areas:

Agriculture	Food Service
Aviation	Health
Building & Construction	Heating & Air Conditioning
Trades	Machine Shop
Clerical Occupations	Management & Supervision
Communications	Meteorology & Navigation
Drafting	Photography
Electronics	Public Service
Engine Mechanics	

The number of courses and the subject areas represented will expand as additional materials with application to vocational and technical education are identified and selected for dissemination.

How Can These Materials Be Obtained?

Contact the Curriculum Coordination Center in your region for information on obtaining materials (e.g., availability and cost). They will respond to your request directly or refer you to an instructional materials agency closer to you.

CURRICULUM COORDINATION CENTERS

EAST CENTRAL

Rebecca S. Douglass
Director
100 North First Street
Springfield, IL 62777
217/782-0759

NORTHWEST

William Daniels
Director
Building 17
Airdustrial Park
Olympia, WA 98504
206/753-0879

MIDWEST

Robert Patton
Director
1515 West Sixth Ave.
Stillwater, OK 74704
405/377-2000

SOUTHEAST

James F. Shill, Ph.D.
Director
Mississippi State University
Drawer DX
Mississippi State, MS 39762
601/325-2510

NORTHEAST

Joseph F. Kelly, Ph.D.
Director
225 West State Street
Trenton, NJ 08625
609/292-6562

WESTERN

Lawrence F. H. Zane, Ph.D.
Director
1776 University Ave.
Honolulu, HI 96822
808/948-7834

Course Description:

This course is designed to provide the skill to program electronic computers structured in Cobol and to code jobstreams using IBM, DOS, JCL, and utility programs. The course covers 8 weeks of instruction.

Data Representation consists of a programmed text covering the binary, octal and hexadecimal number systems.

Basic Cobol Programming section contains a text and problem exercises with some having answers. Topic areas include processing and updating a sequential file; producing an edited report and a report with calculations; processing external and internal tables; debugging syntax errors.

The Operating Systems segment focuses on disk operating systems organization and operation. Specific topics of coding DOS Jobstreams, DOS Librarian Programs, DOS Utilities and Sorts are included.

Programmer/Analyst Course
532-74F1

Classroom Course 4-4

TABLE OF CONTENTS

	<u>Page</u>
Program of Instruction	1
Data Representation Programmed Text	20
Basic Cobol Programming	148
Operating Systems	473

Annex F - Update a Sequential Master File Using a Sequential Transaction File

Terminal Learning Objective: Given program specifications, write a COBOL program to update a sequential master file with a transaction file containing adds, deletes, changes and errors. The program must also produce an update listing and a master file listing.

<u>532-74F-1242</u>	<u>Establish Read Switches</u>	<u>Type</u>
Hours -	2	2C

Learning Objective: Using working storage, establish read switches to govern the access to each file.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1243</u>	<u>Update Master File</u>	<u>Type</u>
Hours -	29	7C, 22PE2

Learning Objective: Given program specifications, match records on two files to produce an updated output master.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1244</u>	<u>COBOL V Review and Exam</u>	<u>Type</u>
Hours -	8	4C, 4E

Learning Objective: N/A

Ref: N/A

Annex F Total Hours Allotted: 39

Annex E - Process External and Internal Tables in COBOL

Terminal Learning Objective: Given program specifications code COBOL routines to build tables from external or internal data and match input files to the completed table.

<u>532-74F-1238</u>	<u>Establish Internal Table</u>	<u>Type</u>
Hours -	14	2C, 12PE2

Learning Objective: Using a month conversion table, write the COBOL statements to change the numeric month to the month's name.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1239</u>	<u>External Table Handling</u>	<u>Type</u>
Hours -	14	5.5C, 8.5PE2

Learning Objective: Given program specifications, code COBOL routines to build a table and search the table for match or no match condition.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1240</u>	<u>COBOL IV Exam and Review/Critique</u>	<u>Type</u>
Hours -	6	2C, 4E

Learning Objectives: N/A

Ref: N/A

Annex E Total Hours Allotted; 34

<u>532-74F-1234</u>	<u>COBOL III Review and Exam</u>	<u>Type</u>
Hours -	8	4C, 4E

Learning Objective: N/A

Ref: N/A

Annex D Total Hours Allotted: 37

Annex D - Produce a Report with Calculations and Minor Control Breaks in COBOL

Terminal Learning Objective: Given program specifications, code a COBOL program that will perform arithmetic calculations and test for changes in input data to produce a report with group totals at a control break.

<u>532-74F-1230</u>	<u>Test for Control Breaks</u>	<u>Type</u>
Hours -	12.5	2.5C, 10PE2

Learning objective: Given input control fields, code routines to process changes in the fields that will result in recognizing the change and processing each group of like items as individual entities.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1232</u>	<u>Advanced Condition Tests</u>	<u>Type</u>
Hours -	2	2C

Learning Objective: Given specifications, code routines that use condition names and nested IF statements syntactically and logically correct.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1232</u>	<u>Arithmetic Verbs</u>	<u>Type</u>
Hours -	6	1.5. 4.5PE2

Learning Objective: Given specifications, code COBOL arithmetic verbs in conjunction with condition statements to produce minor totals at control breaks.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1233</u>	<u>Page Headers & Trailers</u>	<u>Type</u>
Hours -	8.5	2.5C, 6PE2

Learning Objective: Given report requirements, code COBOL routines that will result in multiple headings with the current data and sequential page numbers on output reports.

Ref: Shelley/Cashman Structured COBOL

532-74F-1224COBOL II Review & ExamType

Hours -

6

2C, 4E

Learning Objective: N/A

Ref: N/A

Annex C Total Hours Allotted: 39

6

Annex C - Produce Edited Report with Minor Control Breaks in COBOL

Terminal Learning Objective: Given program specifications, code a COBOL program to read sequential input, perform condition tests, accumulate totals and use report editing to produce a report with a heading and total line.

<u>532-74F-1220</u>	<u>Code Special Names Paragraph</u>	<u>Type</u>
Hours -	.5	.5C

Learning Objective: Given the COBOL manual, code the special names paragraph correctly.

Ref: N/A

<u>532-74F-1221</u>	<u>Establishing a Heading Line and Accumulations in the Data Division</u>	<u>Type</u>
Hours -	1.5	1.5C

Learning Objective: Given program specifications code the Data Division entries to establish a heading line and accumulators.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1222</u>	<u>Code ADD and IF Verbs Logically Within the Procedure Division</u>	<u>Type</u>
Hours -	14	4C, 10PE2

Learning Objective: Given program specifications, code the COBOL routines to print a heading, accumulate totals and to conditional tests.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1223</u>	<u>Editing Data and Formatting A Total Line</u>	<u>Type</u>
Hours -	17	5C, 12PE2

Learning Objective: Given program specifications, write COBOL routines to edit numeric data and print a final total line.

Ref: Shelley/Cashman Structured COBOL

532-74F-1214Code Simple COBOL ProgramType

Hours -

10

1C, 9PE2

Learning Objective: Given program specifications, code a program to read a sequential file, move data and print a simple report.

Ref: Shelly/Cashman Structured COBOL

532-74F-1215COBOL I Exam & Review/CritiqueType

Hours -

7

3C, 4E

Learning Objective: N/A

Ref: N/A

Annex B Total Hours Allotted: 37

Annex B - Process Sequential File in COBOL

<u>532-74F-1210</u>	<u>Code COBOL Identification Division</u>	<u>Type</u>
---------------------	---	-------------

Hours -	1.5	1C, .5PE2
---------	-----	-----------

Learning Objective: Given program specifications, code a COBOL Identification Division that is syntactically correct and that meets specifications.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1211</u>	<u>Code COBOL Environment Division</u>	<u>Type</u>
---------------------	--	-------------

Hours -	1.5	1C, .5PE2
---------	-----	-----------

Learning Objective: Given program specifications, code a COBOL Environment Division that is syntactically correct and that meets specifications.

Ref: Shelley/Cashman Structured COBOL

<u>532-74F-1212</u>	<u>Code COBOL Data Division</u>	<u>Type</u>
---------------------	---------------------------------	-------------

Hours -	8	4C, 4PE2
---------	---	----------

Learning Objective: Given program specifications, code a COBOL Data Division that is syntactically correct and that contains all entries necessary for correct program operation.

Ref: Shelly/Cashman Structured COBOL

<u>532-74F-1213</u>	<u>Code I/O and Movements Verbs</u>	<u>Type</u>
---------------------	-------------------------------------	-------------

Hours -	9	9C
---------	---	----

Learning Objective: Given program specifications, code the COBOL verbs necessary to perform sequential I/O operations and to move data within memory.

Ref: Shelly/Cashman Structured COBOL

<u>532-74F-1205</u>	<u>Problem Solving Exam & Review</u>	<u>Type</u>
---------------------	--	-------------

Hours -	7	3C, 4E
---------	---	--------

Learning Objective: N/A

Ref: N/A

Annex A Total Hours Allotted: 41 9

SECTION IV - ANNEXES

Annex A - Solve ADP Problems

Terminal Learning Objectives: Given a narrative problem, construct a systems flowchart, structured flowchart, HIPO and IPO that will represent a solution to the problem in structured terms.

<u>531-74F-1201</u>	<u>Draw a Systems Flowchart</u>	<u>Type</u>
Hours -	1.5	.5C, 1PE2

Learning Objective: Given the inputs and outputs to a program, draw a chart that will use the proper symbols to depict the flow of data in the system.

Ref: Class Notes

<u>531-74F-1202</u>	<u>Draw a Hierarchy Chart</u>	<u>Type</u>
Hours -	5	3C, 2PE2

Learning Objective: Given program specifications, draw a hierarchy chart that correctly reflects the logical structure of a program that will result in the required output.

Ref: Class Notes, Check Problem

<u>532-74F-1203</u>	<u>Write an IPO Chart</u>	<u>Type</u>
Hours -	15.5	6C, 9.5PE2

Learning Objective: Given a HIPO and program specifications correctly determine the inputs, outputs and processing required to meet the requirements.

Ref: Class Notes, PE 1 through PE 4

<u>532-74F-1204</u>	<u>Draw a Structured Flowchart</u>	<u>Type</u>
Hours -	12	4C, 8PE2

Learning Objective: Given program specifications, a HIPO and an IPO draw a structured flowchart that represents the logic flow within an IPO.

Ref: Class Notes, PE1 through PE4

A. Course: 532-74F1, Programmer/Analyst

B. Purpose: To provide enlisted personnel with the ability to program electronic computers in structured COBOL; to code job streams using IBM DOS JCL and utility programs and to work in the Army ADP environment interfacing with USACSC and other ADP agencies. Personnel are trained in MOS 74F10

C. Prerequisites:

1. Must have a course in high school algebra or score 45 or greater on GED test 5, high school level.

2. Standard score of 105 or better in aptitude area ST or CL.

3. Nine months or more of active duty remaining after course completion.

4. No security clearance required for this course.

5. AR 611-201, Special Requirements Section, requires a security clearance for MOS 74F.

D. Scope: Students learn ADP topics through in-class conference, practical exercises in COBOL, JCL and utilities and self-paced instruction.

E. Length:	Peacetime	Mobilization
	9 weeks, 3 days	8 weeks, 3 days

F. Training Location: US Army Institute of Administration
Fort Benjamin Harrison, Indiana 46216

G. MOS Feeder pattern:

Prerequisite MOS	MOS trained in this course	Feeds Following MOS
09B	74F10	74F20, 74F30, 74F40, 74Z50

H. Ammunition Requirements: None.

UNITED STATES ARMY INSTITUTE OF ADMINISTRATION
Fort Benjamin Harrison, Indiana 46216

PROGRAM OF INSTRUCTION
FOR
PROGRAMMER/ANALYST COURSE
532-74F1

Length: Peacetime - 9 weeks, 3 days

Mobilization - 8 weeks, 3 days

Approved by:

121-012-1400-190-A

Annex G - Data Representation

Terminal Learning Objective: Given the decimal binary and hexadecimal numbering systems, be able to describe the properties and manipulations of each and be able to manipulate the numbers within each number system.

531-74F-1245

Manipulate Number Systems

Type

Hours -

2

1C, 1PE2

Learning Objective: Given several problems in the decimal, binary, and hexadecimal number systems, perform the indicated manipulations for each.

Ref: Data Representation Handout

531-74F-1246

Exam and Review/Critique

Type

Hours -

2

2E

Learning Objective: N/A

Ref: N/A

Annex G Total Hours Allotted: 4

Annex H - Debug COBOL Syntax Errors

Terminal Learning Objective: Given a COBOL listing with three different levels of syntax errors, determine all statements in error and correct them.

<u>531-74F-1247</u>	<u>Determine Erroneous Statement</u>	<u>Type</u>
Hours -	3	1.5C, 1.5PE2

Learning Objective: Given a COBOL diagnostic and source listing, find all language syntax errors and correct them.

Ref: Shelley/Cashman Structured COBOL

<u>531-74F-1248</u>	<u>Exam and Review/Critique</u>	<u>Type</u>
Hours -	1	1E

Learning Objective: N/A

Ref: N/A

Annex H Total Hours Allotted: 4

Annex I - Debug DOS Program Abends

Terminal Learning Objective: Given a COBOL compile listing with all options, a linkedit map and a core dump, find the error causing the program to abend and correct it.

531-74F-1250Code a Simple ALC ProgramType

Hours -

8

6C, 2PE2

Learning Objective: Given in-class notes, code and execute an ALC program to produce a calendar.

Ref: ALC handout

531-74F-1251Correct DOS AbendType

Hours -

10

3C, 7PE2

Learning Objective: Given compile and linkedit listings, and a core dump, find a statement in error and correct it.

Ref: PE handouts

531-74F-1252Exam and Review/CritiqueType

Hours -

4

2C, 2E

Learning Objectives: N/A

Ref: N/A

Annex I Total Hours Allotted: 22

Annex J - Know DOS Organization and Operation

Terminal Learning Objective: Given the System Control and Service Manual, describe the function of each component of DOS as well as the benefits of operating systems and the effects of multiprogramming.

<u>532-74F-1255</u>	<u>Operating Systems</u>	<u>Type</u>
Hours -	2	2C

Learning Objective: Given class reference materials, describe the purpose and advantages of an operating system naming at least 4 advantages.

Ref: OS Handout, System Control and Service

<u>532-74F-1256</u>	<u>Multiprogramming Concepts</u>	<u>Type</u>
Hours -	2	2C

Learning Objective: With class notes, describe the effect multiprogramming has on total system throughput.

Ref: OS Handout

<u>532-74F-1257</u>	<u>DOS Libraries</u>	<u>Type</u>
Hours -	2	2C

Learning Objective: Given class references and a list of programs, state which DOS library will store each program.

Ref: OS Handout
 System Control and Service
 DOS Job Control, Shelly/Cashman

Annex J Total Hours Allotted: 6

Annex K - Code DOS Jobstream

Terminal Learning Objective: Given all class reference material, program specifications and job requirements, code jobstreams to compile link and/or execute programs in the DOS or DOS-E environment.

532-74F-1260Code DOS JobstreamType

Hours -

16.5

9.5C, 7PE2

Learning Objective: Given all class reference material, program specifications and job requirements, code JCL to compile, link and/or execute programs using unit record and utility I/O devices. The JCL must be free of syntax errors and must meet all requirements.

Ref: OS Handout
System Control & Service
DOS Job Control, Shelley/Cashman

532-74F-1261Code DOS-E Extensions to DOSType

Hours -

3.5

2.5C, 1PE2

Learning Objective: Given a DOS Jobstream, revise the JCL to take advantage of all applicable DOS-E (ADAS, DYNAM-T) benefits.

Ref: Instruction Notes

532-74F-1262Exam and Review/CritiqueType

Hours -

6

2C, 4E

Learning Objective: N/A

Ref: N/A

Annex K Total Hours Allotted: 26

6

Annex L - Code DOS Librarian Programs

Terminal Learning Objective: Given all class references, code the jobstreams to display a library directory, display and/or punch a library entry and perform maintenance on any or all DOS libraries. The jobstream must be syntactically correct and must meet stated requirements.

532-74F-1265

DOS Library Service Routines

Type

Hours -

5.5

3C, 2.5PE2

Learning Objective: Given all class references, code a jobstream using SSERV, CSERV, RSERV and/or DSERV without syntax errors and with all stated requirements met.

Ref: OS Handout
System Control & Service
DOS Job Control, Shelley/Cashman

532-74F-1266

DOS Library Maintenance Routines

Type

Hours -

6.5

3.5C, 3PE2

Learning Objective: Given all class references, code a jobstream to perform any MAINT function on any library. The jobstream must be without syntax errors and must meet stated requirements.

Ref: System Control & Service
DOS Job Control, Shelley/Cashman

Annex L Total Hours Allotted: 12

Annex M - Code DOS Utilities and Sorts

Terminal Learning Objective: Given all class references and job requirements, code a jobstream using file to file utilities and the SORT program. The jobstream must be syntactically correct and must produce output that meets stated requirements.

<u>532-74F-1270</u>	<u>DOS File to File Utilities</u>	<u>Type</u>
Hours -	6	3C, 3PE2

Learning Objective: Given all class references and job requirements, code a jobstream to transfer data from one storage media to another. The jobstream must meet requirements and be syntactically correct.

Ref: DOS & TOS Utilities
OS Handout

<u>532-74F-1270</u>	<u>DOS SORT/MERGE</u>	<u>Type</u>
Hours -	9	3C, 6PE2

Learning Objective: Given all class references and job requirements, code a jobstream to sort records into a given sequence. The jobstream must result in sorted output and must be syntactically correct.

Ref: DOS Tape and Dist SORT/MERGE Program

<u>532-74F-1271</u>	<u>CSC Utilities</u>	<u>Type</u>
Hours	1	1C

Learning Objective: Given the CSC Utilities Manual index, find the desired utility within 3 minutes.

Ref: CSC Utilities Manual

<u>532-74F-1272</u>	<u>Utilities Review & Exam</u>	<u>Type</u>
Hours -	6	2C, 4E

Learning Objective: N/A

Ref: N/A

Annex M Total Hours Allotted: 22

Annex N - Report Systems Problems to USACSC

532-74F-1275Complete an Incident ReportType

Hours -

1.5

1.5PI

Learning Objective: Given a scenario and blank forms, complete an Incident Report (USACSC Form 53) with all required information.

Ref: DAP Interface With USACSC

532-74F-1276Complete a System Change RequestType

Hours -

1.5

1.5PI

Learning Objective: Given a scenario and blank forms, complete a Systems Change Request with all required information.

Ref: DPA Interface With USACSC

532-74F-1277CSC/DPA Interface ExamType

Hours -

1

1E

Learning Objective: N/A

Ref: N/A

Annex N Total Hours Allotted: 4

Annex 0 - Produce Systems Documentation

Terminal Learning Objective: Given sample DOD documentation standards and program specifications, compile an ADP documentation packet in the proper format and with all required information.

532-74F-1280

Programmer's Documentation

Type

Hours -

6

3C, 3PE2

Learning Objective: Given sample DOD standards and applicable specifications, compile program documentation in the right format and with all required information.

Ref: DOD Documentation Standards

532-74F-1281

Operations Documentation

Type

Hours -

5

3C, 2PE2

Learning Objective: Given sample DOD standards and applicable specifications, compile operations documentation in the right format and with all required information.

Ref: DOD Documentation Standards

532-74F-1282

Documentation Review and Exam

Type

Hours -

4

2C, 2E

Learning Objective: N/A

Ref: N/A

Annex 0 Total Hours Allotted: 15

BIN D-024

DATA REPRESENTATION

PART I

BINARY NUMBER SYSTEM

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURES THAT SHOULD BE FOLLOWED:

Read the frame very carefully.

In some cases the frame should be read twice.

Most of the frames require a response on the part of the student.

Write down the answer that you feel is correct before you turn the page to the answer.

The answer will always be found at the top of the next page.

If your answer is correct continue to the next frame.

If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the Binary Number System. Upon completion of the text, you will be able to:

1. Convert a Binary number to a Decimal number.
2. Convert a Decimal number to a Binary number.
3. Convert a Binary Coded Decimal to a Decimal number.
4. Convert a Decimal number to a Binary Coded Decimal.
5. Perform Binary arithmetic functions.

The number system taught in grade school is the Decimal Number System.
In this number system ten symbols are used to represent digits.

What are the ten symbols used to indicate digits in the Decimal Number System?

24

ANS: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9

I hope you have not forgotten the Decimal Number System.

The BASE of a number system is the number of symbols used to indicate digits.

What is the BASE of the Decimal Number System?

ANS. Since ten symbols are used to represent the digits 0 through 9, the BASE of the Decimal Number System is TEN.

The positional values for three positional columns of the Decimal Number System are:

100	10	1

The positional values of any number system are determined by taking the BASE of the number system and raising it to progressive powers substantially.

Progressive
Powers

10^2	10^1	10^0 *

 =

100	10	1

*Any number raised to the zero power is 1.

Indicate the positional values for four columns of the Decimal Number System.

26
ANS.

1000	100	10	1

or

10^3	10^2	10^1	10^0

The Decimal Number 25 represented with the positional value of columns.

10	1
2	5

Indicate the positional values over the Decimal Number 425.

ANS.

100	10	1
4	2	5

By using a combination of positional values in the columns, a number can be represented. To determine the value of the number represented, the digit indicated in a column is multiplied by the positional value of the column; then, the values of the digits are added. For example:

The Decimal Number 425.

100	10	1
4	2	5

←----- Positional values of columns

↓

5 X 1 = 5

2 X 10 = 20

4 X 100 = 400

425

Indicate how the value of Decimal Number 742 is determined.

28
ANS.

1000	100	10	1	
	7	4	2	
				2 X 1 = 2
				4 X 10 = 40
				7 X 100 = 700
				742

The largest number that can be indicated in one column of any number system will be the BASE of the number system minus 1.

What is the largest number that can be indicated in one column of the Decimal Number System?

10	1
9	9

The electronic circuitry used in data processing systems function in binary states. This means that the circuitry can indicate only two possible states, either an "on" or "off" condition. This characteristic made it necessary to develop a number system that could be used by a computer. This number system is called the Binary Number System.

In this number system only the symbols 0 and 1 are used.

What is the BASE of the Binary Number System?

ANS: 2

The BASE of a number system is the number of symbols used to indicate digits. In the Binary Number System only two symbols, 0 and 1, are used.

If the BASE of the Binary System is TWO, the positional values for four columns are:

2^3	2^2	2^1	2^0

or

8	4	2	1

What are the positional values for seven columns of the Binary Number System?

32
ANS:

2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

or	64	32	16	8	4	2	1

What is the largest digit that can be indicated in one positional column in the Binary Number System?

43

$$\begin{array}{rcl}
 \text{ANS:} & 1 & \text{BASE of Binary Number System} & = & 2 \\
 & & \text{Minus 1} & = & \underline{-1} \\
 & & & & 1
 \end{array}$$

As in any number system, using a combination of positional values in the columns a number can be constructed or represented. A combination of 1's and 0's in this number system can represent a decimal number./

Method of converting Binary number to Decimal number.

Example is of Binary number 101 converted to a Decimal number.

1. Assign positional values of Binary Number System above the Binary number.

64	32	16	8	4	2	1
				1	0	1

"1" = "on" condition
 "0" = "off" condition

2. Multiply digit indicated in each column by its positional value.

64	32	16	8	4	2	1
				1	0	1

Positional values of columns

1 X 1 = 1

1 X 4 = 4

3. Add positional values of columns.

$$1 \times 1 = 1$$

$$1 \times 4 = 4$$

5

From this conversion example we have learned that the Binary number 101 is equivalent to the Decimal number 5. Dots under the Binary number indicate positional columns. The columns are represented internally in a computer with a bit that can be turned "on" or "off". So the dot may also represent a bit.

Convert the Binary number 1000111 to a Decimal number.

36

ANS: 85

64	32	16	8	4	2	1
1	0	1	0	1	0	1

$$\begin{array}{rcl}
 1 \times 1 & = & 1 \\
 1 \times 4 & = & 4 \\
 1 \times 16 & = & 16 \\
 1 \times 64 & = & 64
 \end{array}$$

85 (Decimal)
(Number)

Convert Binary 0111001 to a Decimal Number.

ANS: 57

64	32	16	8	4	2	1
0	1	1	1	0	0	1

$$\begin{aligned}
 1 \times 1 &= 1 \\
 1 \times 8 &= 8 \\
 1 \times 16 &= 16 \\
 1 \times 32 &= 32
 \end{aligned}$$

57 (Decimal)
(Number)

TEST

Convert following Binary numbers to Decimal. Each dot (.) under the Binary number indicates a positional column.

	<u>BINARY</u>		<u>DECIMAL</u>
a.	101 ...	=	_____
b.	10001	=	_____
c.	1001	=	_____
d.	111001	=	_____

Method to convert Decimal number to Binary number.

EXAMPLE: Decimal number 91 converted to a Binary number.

1. Assign positional values of the number system the Decimal number is to be converted to. The highest positional value must be larger than the Decimal number to be converted.

Decimal number
to be converted

91

128	64	32	16	8	4	2	1

2. Indicate in the column the number of times the positional value will go into the Decimal number. "0" indicates positional value will not go into Decimal number. Begin with highest positional value.

91	128	64	32	16	8	4	2	1
	0	1						

3. Multiply the positional value by the digit indicated in the column; subtract result from Decimal number.

128	64	32	16	8	4	2	1
0	1						

$$\begin{array}{r} 91 \\ 64 \\ \hline 27 \end{array}$$

◀ (64 X 1)

remainder

4. Indicate the number of times the next positional value will go into the remainder.

128	64	32	16	8	4	2	1
0	1	0	1	1	0	1	1

$$\begin{array}{r} 91 \\ 64 \\ \hline 27 \\ 16 \\ \hline 11 \\ 8 \\ \hline 3 \\ 2 \\ \hline 1 \\ 1 \\ \hline 0 \end{array}$$

◀ (64 X 1)

◀ (16 X 1)

◀ (8 X 1)

◀ (2 X 1)

◀ (1 X 1)

The Decimal number 91 is equivalent to the Binary number 1011011.

Convert the Decimal number 14 to a Binary number.

40.

ANS: 01110
.....

Solution:

16	8	4	2	1
0	1	1	1	0

$$\begin{array}{r} 14 \\ 8 \\ \hline 6 \end{array} \triangleleft (8 \times 1)$$

$$\begin{array}{r} 4 \\ \hline \end{array} \triangleleft (4 \times 1)$$

$$\begin{array}{r} 2 \\ 2 \\ \hline 0 \end{array} \triangleleft (2 \times 1)$$

Convert the Decimal number 72 to a Binary number.

ANS: 01001000
.....

41

Convert the Decimal number 45 to a Binary number.

52

TEST

Convert following Decimal numbers to Binary.

	<u>Decimal</u>		<u>Binary</u>
a.	4	=
b.	15	=
c.	26	=
d.	63	=

BINARY CODED DECIMAL

A Binary number like 01000011011101100101 may be difficult to interpret because of the long string of 1's and 0's. To convert this Binary number to Decimal would involve assigning positional values to 20 positional columns, then adding their positional values. This would take a lot of time! For readability purposes and to save time, the Binary Coded Decimal was designed to easily convert a Binary number to a Decimal number. In Binary Coded Decimal (BCD) four bits are used to indicate one Decimal number. The positional values of these four bits are the same as the first four bits in the Binary Number System.

What are the positional values of the four bits in BCD?

44

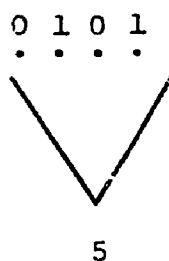
ANS:

8	4	2	1

The Decimal number 5 represented with four bits in Binary Coded Decimal (BCD).

8	4	2	1
0	1	0	1

thus

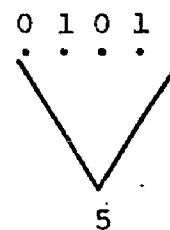
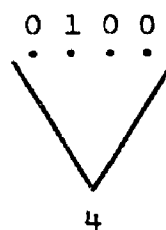


The Decimal number 45 represented in BCD:

8	4	2	1
0	1	0	0

8	4	2	1
0	1	0	1

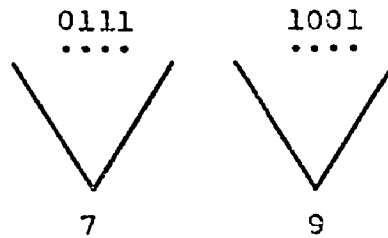
thus,



In BCD only the digits 0 through 9 can be indicated in one group of four bits.

Convert the Decimal number 79 to BCD.

ANS: 0111 1001



Convert the Decimal number 458 to BCD.

46

ANS:

0100
....

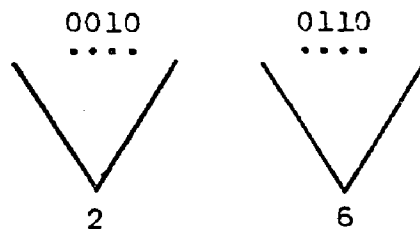
0101
....

1000
....

Convert BCD 0010 0110 to a Decimal number.

ANS: 26

47



Convert BCD 0111 0011 0100 1001 to a Decimal number.

0111 0011 0100 1001
....

7 3 4 9

TEST

1. Convert Binary Coded Decimal (BCD) to Decimal.

	<u>8421</u>	<u>8421</u>	<u>8421</u>	<u>100</u>	<u>10</u>	<u>1</u>
a.			0011
b.	0010	0010
c.	0101	0100
d.	0111	0110	1000

2. Convert Decimal to BCD.

	<u>100</u>	<u>10</u>	<u>1</u>	<u>8421</u>	<u>8421</u>	<u>8421</u>
a.		7			
b.		1	5	
c.		7	6	
d.	4	2	3



BINARY ADDITION

Rules applied when performing Binary addition:

a. $0 + 0 = 0$

b. $1 + 0$ or $0 + 1 = 1$

c. $1 + 1 = 0$ and carry 1 to next column to left

d. $1 + 1 + 1 = 1$ and carry 1 to next column to left

Addition problem: $\begin{array}{r} 010 \\ + 011 \\ \hline \end{array}$

$\begin{array}{r} 1 \\ + 010 \\ + 011 \\ \hline 101 \end{array}$
1 carried
same as
 $\begin{array}{r} 2 \\ + 3 \\ \hline 5 \end{array}$

Add: $\begin{array}{r} 101 \\ + 101 \\ \hline \end{array}$

50
ANS: 1010

Solution:
$$\begin{array}{r} 1\ 1 \\ 101 \\ + \quad \underline{101} \\ 1010 \end{array}$$

Add: 001
+ 111

ANS: 1000

51

Add: 0110
+ 1100

52

ANS:

10010

Add: 0110101
+ 1110100

TEST

Perform Binary addition

a. 101
 + 001

...

b. 1000
 + 1001

.....

c. 1011
 + 1001

.....

d. 01110
 11011

.....

BINARY SUBTRACTION

The method by which subtraction is performed internally in a computer is Subtraction by complementation. The complement of a number is the amount that must be added to a certain number to have it indicate the largest number that can be indicated in one positional column. In Binary the complement of 1 is 0, the complement of 0 is 1 and, really, all we are doing is reversing the binary notation of the subtrahend (the number that is to be subtracted from another).

Subtraction problem:
$$\begin{array}{r} 101 \\ - 011 \\ \hline \end{array}$$

- a. Place the proper sign bit before each binary number; 0 = plus (+), 1 = minus (-). A plus sign is applied to minuend (top number) when minuend does not have plus or minus sign.

$$\begin{array}{r} 101 \\ - 011 \\ \hline \end{array} \quad \begin{array}{c} \text{Sign bits} \\ \swarrow \\ \begin{array}{|c|c|c|} \hline 0 & 101 \\ \hline 1 & 011 \\ \hline \end{array} \end{array}$$

- b. Complement subtrahend and add. Do not complement minuend or sign bits.

$$\begin{array}{|c|c|c|} \hline 0 & 101 \\ \hline 1 & 011 \\ \hline \end{array} \xrightarrow{\text{(Subtrahend)}} \begin{array}{|c|c|c|} \hline 0 & 101 \\ \hline 1 & 100 \\ \hline 1 & 0 & 001 \\ \hline \end{array}$$

Anything to left of sign bit is dropped or truncated. Thus 1 0 001 becomes 0 001.

- c. Add 1 to result just obtained.

$$\begin{array}{|c|c|c|} \hline 0 & 001 \\ \hline 1 & \\ \hline 0 & 010 \\ \hline \end{array}$$

Sign bit is 0 or +. Thus answer is + 2.

Subtract:
$$\begin{array}{r} 100 \\ - 010 \\ \hline \end{array}$$

56

ANS:

0 010

Solution:

sign bit

$$\begin{array}{r}
 \begin{array}{r}
 100 \\
 - 010
 \end{array}
 \left. \vphantom{\begin{array}{r} 100 \\ - 010 \end{array}} \right\}
 \begin{array}{r}
 0 \quad 100 \\
 1 \quad 101
 \end{array}
 \\
 \hline
 1 \quad 0 \quad 001
 \end{array}
 =
 \begin{array}{r}
 0 \quad 001 \\
 + 1 \\
 \hline
 0 \quad 010 \quad \text{or } +2
 \end{array}$$

Subtract: 1000
 - 0111

67

ANS:

0	0001
---	------

Subtract: 10110
- 01101

58

ANS:

9

01001

TEST

Subtract:

$$\begin{array}{r} \text{a. } 0111 \\ - 0101 \\ \hline \end{array}$$

$$\begin{array}{r} \text{b. } 1000 \\ - 0100 \\ \hline \end{array}$$

$$\begin{array}{r} \text{c. } 1001 \\ - 0110 \\ \hline \end{array}$$

$$\begin{array}{r} \text{d. } 10001 \\ - 01010 \\ \hline \end{array}$$

DATA REPRESENTATION

PART II

OCTAL NUMBER SYSTEM

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURE THAT SHOULD BE FOLLOWED:

Read the frame very carefully.

In some cases the frame should be read twice.

Most of the frames require a response on the part of the student.

Write down the answer that you feel is correct before you turn the page to the answer.

The answer will always be found at the top of the next page.

If your answer is correct continue to the next frame.

If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the Octal Number System. Upon completion of the text, you will be able to:

1. Convert an Octal number to a Decimal number.
2. Convert a Decimal number to an Octal number.
3. Convert Binary Coded Octal to a Decimal number.
4. Convert a Decimal number to Binary Coded Octal.
5. Perform Octal arithmetic functions.

OCTAL NUMBER SYSTEM

In the Octal Number System EIGHT symbols are used to represent its digits.

What is the BASE of the Octal Number System?

ANS: The BASE of the Octal Number System is EIGHT. This is because eight symbols are used to indicate digits in this number system.

Since the BASE of the Octal Number System is EIGHT, the positional values of the columns will be the BASE of the number system raised to progressive powers sequentially.

What is the positional value of the first column in the Octal Number System?

64

ANS: The first column will always be the BASE of the number system raised to the zero power. Any number to the zero power is one. Thus; First column = 8^0 or 1, which is indicated as

8^0	or	1

The positional values of three columns of the Octal number system are:

8^2	8^1	8^0	or	64	8	1

What are the positional values for four columns of the Octal number system?

75

ANS:

8	3	8	2	8	1	8	°	512	64	8	1

or

The largest number that can be indicated in one column of any number system will be the BASE of the number system minus 1.

What is the largest number that can be indicated in one column of the Octal Number System?

66

ANS: 7

$$\begin{array}{rcl} \text{BASE of Octal Number System} & = & 8 \\ \text{Minus 1} & = & - \frac{1}{7} \end{array}$$

If the largest number that can be indicated in each column is 7, what are the symbols to indicate digits in the Octal Number System?

ANS: 0, 1, 2, 3, 4, 5, 6, and 7

Method for converting OCTAL number system to DECIMAL number.

Example of the OCTAL number 225 converted to a DECIMAL number.

1. Assign positional values of OCTAL number system above the OCTAL number.

512	64	8	1
	2	2	5

2. Multiply digit indicated in each column by its positional value.

512	64	8	1
	2	2	5

Positional values of columns

$$\begin{array}{l}
 \downarrow \\
 5 \times 1 = 5 \\
 2 \times 8 = 16 \\
 2 \times 64 = 128
 \end{array}$$

3. Add positional values of columns

$$\begin{array}{rcl} 5 \times 1 & = & 5 \\ 2 \times 8 & = & 16 \\ 2 \times 64 & = & 128 \\ \hline & & 149 \end{array}$$

Thus, the OCTAL number 225 is equivalent to the DECIMAL number 149.

Convert the OCTAL number 136 to a DECIMAL number.

ANS: DECIMAL 94

Solution:

64	8	1
1	3	6
1	1	1

$$6 \times 1 = 6$$

$$3 \times 8 = 24$$

$$1 \times 64 = \underline{64}$$

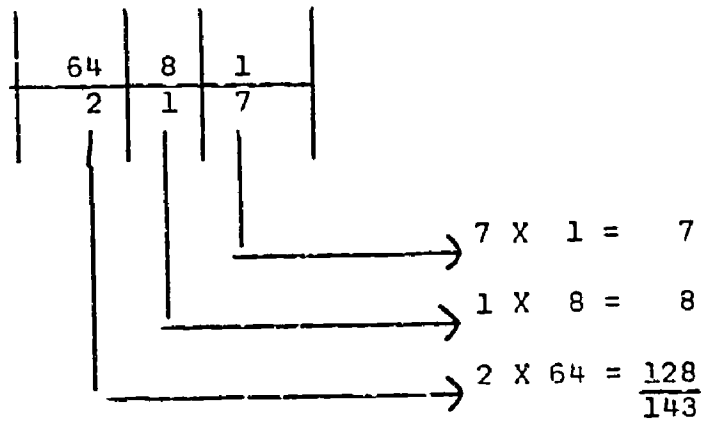
94

Convert the Octal number 217 to a Decimal number.

70

ANS: DECIMAL 143

Solution:

TEST

Convert OCTAL number to DECIMAL number

	<u>8</u>	<u>1</u>		<u>10</u>	<u>1</u>
a.	1	2		.	.
b.	1	7		.	.
c.	3	2		.	.
d.	7	6		.	.

STEPS TO CONVERT DECIMAL NUMBER TO OCTAL NUMBER.

The Decimal number 89 converted to an Octal number.

1. Assign positional values of Octal Number System. The highest positional value must be larger than Decimal Number to be converted.

					Highest Positional Value
512	64	8	1		

Decimal Number to be
Converted

↓
89

2. Indicate in the column the number of times the positional value will go into the Decimal number. "0" indicates positional value will not go into Decimal number. (Begin with highest positional value)

512	64	8	1	
0				

89

3. If positional value in the column will go into the Decimal number, multiply the positional value that goes into the Decimal number by the digit indicated in the column; then, subtract result from the Decimal number.

512	64	8	1
0	1		

$$\begin{array}{r} 89 \\ 64 \\ \hline 25 \end{array}$$

1×64

4. Indicate in the following columns the number of times the positional values will go into remainder until remainder is zero.

512	64	8	1
0	1	3	1

$$\begin{array}{r} 89 \\ 64 \\ \hline 25 \\ 24 \\ \hline 1 \\ 1 \\ \hline 0 \end{array}$$

1×64
 3×8
 1×1

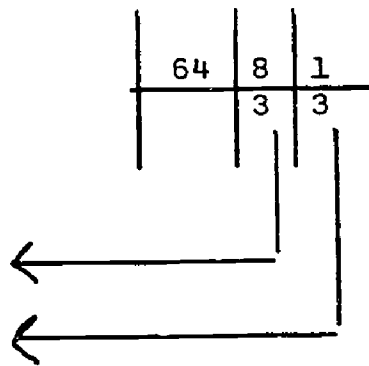
Thus, the Decimal number 89 is equivalent to the Octal number 131.

Convert the Decimal number 27 to an Octal number.

ANS: Octal 33

Solution:

$$\begin{array}{r}
 27 \\
 24 \\
 \hline
 3 \\
 3 \\
 \hline
 0
 \end{array}$$



Convert the Decimal number 42 to an Octal number.

4

: Octal 52

Solution:

64	8	1
	5	2

42

40

2

2

0

←

←

ANS: OCTAL 315

Solution:

	512	64	8	1
	0	3	1	5
205				
192				
13				
8				
5				
5				
0				

(3×64)
 (1×8)
 (5×1)

TEST

Convert Decimal to Octal

	10	1		8	1
a.		6			.
b.	1	8		.	.
c.	4	6		.	.
d.	6	3		.	.

BINARY CODED OCTAL (BCO)

Binary Coded Decimal one decimal number was represented with 1's and 0's in a group of four bits (four columns). In Binary Coded Octal three bits are used to represent one Octal number with 1's and 0's. The positional values of these three bits are the same as the first three bits in the Binary Numbering System.

What are the positional values of the three bits in Binary Coded Octal?

ANS:

4	2	1

The OCTAL number 6 is represented with three bits in Binary Coded Octal:

4	2	1
1	1	0

thus,

1 1 0

6

The OCTAL number 36 represented in Binary Coded Octal.

4	2	1
0	1	1

4	2	1
1	1	0

thus,

0 1 1

1 1 0

3

6

In Binary Coded Octal only the digits 0 through 7 can be indicated in one group of three bits.

Convert the Binary Coded Octal number 0 1 0 1 0 0 to an Octal number.

3
ANS: 24

Solution:

$\begin{array}{ccc} 0 & 1 & 0 \\ \cdot & \cdot & \cdot \\ \hline \end{array}$
2

$\begin{array}{ccc} 1 & 0 & 0 \\ \cdot & \cdot & \cdot \\ \hline \end{array}$
4

Convert the Binary Coded Octal number $\begin{array}{ccc} 1 & 0 & 1 \\ \cdot & \cdot & \cdot \end{array} \begin{array}{ccc} 0 & 0 & 1 \\ \cdot & \cdot & \cdot \end{array}$ to an Octal number.

ANS: 51

Solution:

$$\begin{array}{ccc} 1 & 0 & 1 \\ \hline & & \\ \hline & & \end{array} \quad \begin{array}{ccc} 0 & 0 & 1 \\ \hline & & \\ \hline & & \end{array}$$

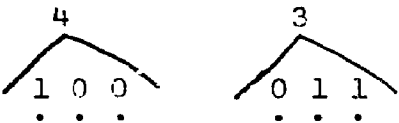
5 1

Convert OCTAL number 43 to Binary Coded Octal.

80.

ANS: 1 0 0 0 1 1

Solution

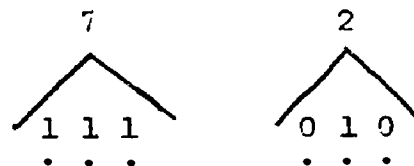


Convert OCTAL number 72 to Binary Coded Octal.

9.

ANS: 1 1 1 0 1 0

Solution



TEST

1. Convert Binary Coded Octal to Octal number.

	4 2 1	4 2 1	8 1	
a.	0 0 1	0 0 1	. .	(Octal number)
b.	0 0 1	1 0 1	. .	(Octal number)
c.	1 0 1	1 1 0	. .	(Octal number)
d.	1 1 1	1 0 0	. .	(Octal number)

2. Convert Octal number to Binary Coded Octal.

	64 8 1	4 2 1	4 2 1	4 2 1	
a.	5				. . .
b.	2 3				. . .
c.	5 2				. . .
d.	1 2 7				. . .

OCTAL ADDITION:

$$\begin{array}{r} 43 \\ + 56 \\ \hline \end{array}$$

1. Add one column at a time.

$$\begin{array}{r} \boxed{43} \\ + \boxed{56} \\ \hline 9 \end{array}$$

2. If result is 7 or less indicate result in column. But if result is more than 7, subtract result by 8 (the base of the Octal number system).

$$\begin{array}{r} \boxed{43} \\ + \boxed{56} \\ \hline \end{array} \quad \begin{array}{r} 3 \\ +6 \\ \hline 9 \\ -8 \leftarrow \\ \hline 1 \end{array}$$

3. Indicate remainder in the column added and carry 1 to next column.

First column:

$$\begin{array}{r}
 \text{1 carry} \rightarrow \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 4 & 6 \\ \hline 5 & 6 \\ \hline 1 & \\ \hline \end{array}
 \end{array}
 \quad
 \begin{array}{r}
 3 \\
 + 6 \\
 \hline
 9 \\
 - 8 \\
 \hline
 1
 \end{array}$$

4. Add following columns in same manner (include 1 carry).

Second column:

$$\begin{array}{r}
 \text{1 carry} \rightarrow \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 4 & 6 \\ \hline 5 & 6 \\ \hline 2 & 1 \\ \hline \end{array}
 \end{array}
 \quad
 \begin{array}{r}
 1 \\
 4 \\
 5 \\
 \hline
 10 \\
 - 8 \\
 \hline
 2
 \end{array}$$

Third column:

$$\begin{array}{|c|c|} \hline 1 & 1 \\ \hline & 4 \ 3 \\ \hline & 5 \ 6 \\ \hline 1 & 2 \ 1 \\ \hline \end{array}$$

Thus, the result of the OCTAL addition is 121.

Perform OCTAL addition:

$$\begin{array}{r}
 2 \\
 + 7 \\
 \hline
 \end{array}$$

9.1

84

ANS: 11

Perform OCTAL addition:

$$\begin{array}{r} 42 \\ + 55 \\ \hline \end{array}$$

95

ANS: 117

TEST

Perform OCTAL addition:

a.
$$\begin{array}{r} 2 \\ + 5 \\ \hline \end{array}$$

b.
$$\begin{array}{r} 12 \\ + 7 \\ \hline \end{array}$$

c.
$$\begin{array}{r} 52 \\ + 66 \\ \hline \end{array}$$

d.
$$\begin{array}{r} 64 \\ + 31 \\ \hline \end{array}$$

OCTAL SUBTRACTION

$$\begin{array}{r} 45 \\ - 17 \\ \hline \end{array}$$

1. Subtract one column at a time.

$$\begin{array}{r} \boxed{4 \ 5} \\ - \boxed{1 \ 7} \\ \hline \end{array} \quad \begin{array}{r} \swarrow \\ 5 \\ - 7 \\ \hline \end{array}$$

2. If minuend (number to be subtracted from) is smaller than subtrahend, borrow the base (8) from next column.

$$\begin{array}{r} 3 \quad \boxed{4 \ 5} \\ \swarrow \quad - \boxed{1 \ 7} \\ \hline \end{array} \quad \begin{array}{r} \swarrow \\ 1 \ 3 \\ - 7 \\ \hline \end{array}$$

(5 + 8 (Base))

3. Indicate result in the column subtracted

First Column:

$$\begin{array}{r}
 3 \quad \boxed{5} \quad 1 \quad 3 \\
 \cancel{4} \quad \boxed{7} \quad - \quad 7 \\
 \hline
 6 \quad \quad \quad 6
 \end{array}$$

Diagram illustrating the first column subtraction in octal. A box is drawn around the digits 5 and 7. An arrow points from the 3 in the next column to the 5, indicating a borrow. The result 6 is written below the 5 and 7.

4. Subtract following column in same manner

Second Column:

$$\begin{array}{r}
 3 \quad \boxed{5} \quad 3 \\
 \cancel{4} \quad \boxed{7} \quad - \quad 1 \\
 \hline
 2 \quad 6 \quad \quad 2
 \end{array}$$

Diagram illustrating the second column subtraction in octal. A box is drawn around the digits 5 and 7. An arrow points from the 3 in the next column to the 5, indicating a borrow. The result 2 is written below the 5 and 7.

Thus, the result of the OCTAL subtraction is 26.

Perform OCTAL subtraction:

$$\begin{array}{r}
 34 \\
 - 15 \\
 \hline
 \end{array}$$

88

ANS: 17

Perform OCTAL subtraction:

$$\begin{array}{r} 73 \\ - 35 \\ \hline \end{array}$$

90

ANS: 36

TEST

Perform OCTAL subtraction:

$$\begin{array}{r} 52 \\ \text{a. } -14 \\ \hline \end{array}$$

$$\begin{array}{r} 64 \\ \text{b. } -25 \\ \hline \end{array}$$

$$\begin{array}{r} 37 \\ \text{c. } -17 \\ \hline \end{array}$$

$$\begin{array}{r} 66 \\ \text{d. } -57 \\ \hline \end{array}$$

DATA REPRESENTATION
PART III
HEXADECIMAL NUMBER SYSTEM

91
READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURE THAT SHOULD BE FOLLOWED:

Read the frame very carefully.

In some cases the frame should be read twice.

Most of the frames require a response on the part of the student.

Write down the answer that you feel is correct before you turn the page to the answer.

The answer will always be found at the top of the next page.

If your answer is correct continue to the next frame.

If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the Hexadecimal Number System. Upon completion of the text, you will be able to:

1. Convert a Hexadecimal number to a Decimal number.
2. Convert a Decimal number to a Hexadecimal number.
3. Convert Binary Coded Hexadecimal to a Decimal number.
4. Convert a Decimal number to Binary Coded Hexadecimal.
5. Perform Hexadecimal arithmetic functions.

HEXADECIMAL NUMBER SYSTEM

In the Hexadecimal Number System SIXTEEN symbols are used to represent it's digits.

What is the BASE of the Hexadecimal Number System?

94

ANS: The BASE of the HEXADECIMAL number system is SIXTEEN. This is because sixteen symbols are used to represent digits in this number system.

Since the BASE of the HEXADECIMAL number system is SIXTEEN, the positional values of the columns will be the BASE of the number system raised to progressive powers sequentially.

What is the positional value of the first column in HEXADECIMAL number system?

ANS:

16^0

 or

1

The first column will always be the BASE of the number system raised to the zero power. Any number to the zero power is one. Thus first column = 16^0 or 1.

The positional values of three columns of the HEXADECIMAL number system are:

16^2	16^1	16^0
--------	--------	--------

 or

256	16	1
-----	----	---

What are the positional values for four columns of the HEXADECIMAL number system?

96

ANS:

16^3	16^2	16^1	16^0

or

4096	256	16	1

The largest number that can be indicated in one column of any numbering system will be the BASE of the numbering system minus 1.

What is the largest number that can be indicated in one column of the HEXADECIMAL number system?

ANS: 15 Solution:

$$\begin{array}{rcl} \text{BASE of Hexadecimal Number System} & = & 16 \\ \text{Minus 1} & = & - \frac{1}{15} \end{array}$$

In the HEXADECIMAL number system alphabetic letters are used to indicate 10 through 15.

10	=	A
11	=	B
12	=	C
13	=	D
14	=	E
15	=	F

If the largest digit that can be indicated in each column is F (which is 15), what are the symbols that are used to indicate digits in the Hexadecimal number system?

ANS: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

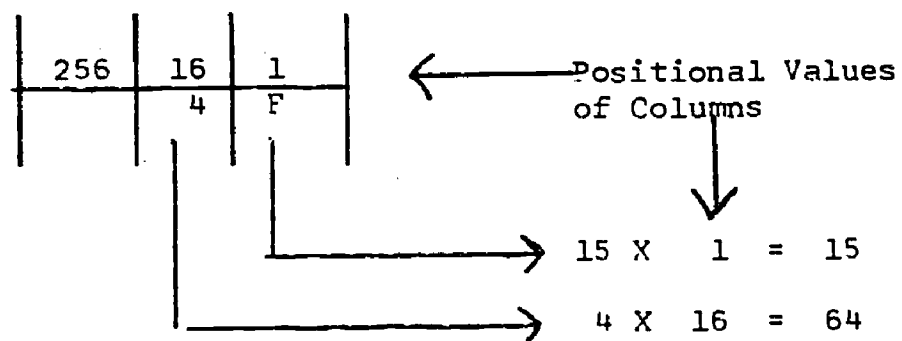
Method for converting HEXADECIMAL number system to DECIMAL number system.

Example of the HEXADECIMAL number 4F converted to a DECIMAL number.

1. Assign positional values of HEXADECIMAL number system above the HEXADECIMAL number.

256	16	1
	4	F

2. Multiply digit indicated in each column by its positional value.



3. Add positional values of columns

$$\begin{array}{r} 15 \times 1 = 15 \\ 4 \times 16 = 64 \\ \hline 79 \end{array}$$

Thus, the HEXADECIMAL number 4F is equivalent to the DECIMAL number 79.

Convert HEXADECIMAL 1B to a DECIMAL number.

110

100

ANS: 27

Solution:

256	16	1
	1	3

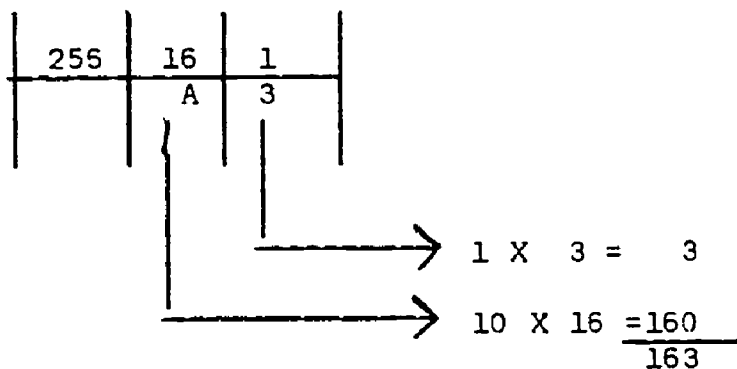
$$\begin{array}{l}
 \rightarrow 11 \times 1 = 11 \\
 \rightarrow 1 \times 16 = \underline{16} \\
 \hline
 27
 \end{array}$$

Convert HEXADECIMAL A3 to a DECIMAL number?

111

ANS: 163

Solution:



TEST

Convert HEXADECIMAL to DECIMAL.

	256	16	1		100	10	1
a.			E			.	.
b.		A	9		.	.	.
c.		F	3		.	.	.
d.	1	C	6		.	.	.

Method for converting DECIMAL number system to HEXADECIMAL number system.

Example of the DECIMAL number 167 converted to a HEXADECIMAL number.

1. Assign positional values of HEXADECIMAL number system. The highest positional value must be larger than DECIMAL number to be converted.

Decimal number
to be converted



167

256	16	1

2. Indicate in the column the number of times the positional value will go into the DECIMAL number. "0" indicates positional value will not go into DECIMAL number (Begin with highest positional value.)

167

256	16	1
0		

3. If positional value in the column will go into the DECIMAL number, multiply the positional value that goes into the DECIMAL number by the digit indicated in the column; then, subtract result from the DECIMAL number.

256	16	1
0	A	

$$\begin{array}{r} 167 \\ 160 \\ \hline 7 \end{array}$$

←

10×16

4. Indicate in the following columns the number of times the positional values will go into remainder until remainder is zero.

256	16	1
0	A	7

$$\begin{array}{r} 167 \\ 160 \\ \hline 7 \\ 7 \\ \hline 0 \end{array}$$

←

10×16

←

7×1

Thus the DECIMAL number 167 is equivalent to the HEXADECIMAL number A7.

Convert the DECIMAL number 230 to a HEXADECIMAL number.

104

ANS: E6

Solution:

	256	16	1
	0	E	6

230

224

6

6

0

14 X 16

X 1

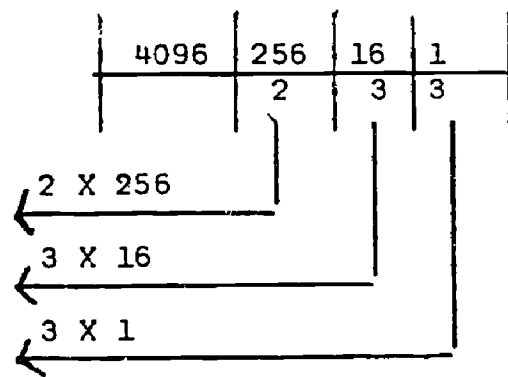
Convert the DECIMAL number 563 to a HEXADECIMAL number.

115

ANS: 233

Solution:

$$\begin{array}{r} 563 \\ 512 \\ \hline 51 \\ 48 \\ \hline 3 \\ 3 \\ \hline 0 \end{array}$$



TEST

Convert Decimal to Hexadecimal

1000	100	10	1
------	-----	----	---

256	16	1.
-----	----	----

- | | | | | | | |
|----|---|---|---|---|---|---|
| a. | | 1 | 3 | | . | . |
| b. | 1 | 7 | 5 | | . | . |
| c. | 9 | 4 | 0 | | . | . |
| d. | 2 | 6 | 1 | 4 | | . |

BINARY CODED HEXADECIMAL (BCH)

In Binary Coded Decimal, one decimal number was represented with 1's and 0's in a group of four bits (four columns). In Binary Coded Hexadecimal, again four bits are used to represent one Hexadecimal number with 1's and 0's. The positional values of these four bits are the same as the first four bits in the Binary Numbering System.

What are the positional values of the four bits in the Binary Coded Hexadecimal?

ANS:

8	4	2	1

The Hexadecimal number A (10) represented in Binary Coded Hexadecimal (with four bits):

8	4	2	1
1	0	1	0

thus,

1 0 1 0
• • • •

A

The Hexadecimal number 6A represented in Binary Coded Hexadecimal.

8	4	2	1
0	1	1	0

8	4	2	1
1	0	1	0

thus,

0 1 1 0
• • • •

6

1 0 1 0
• • • •

A

In Binary Coded Hexadecimal the digits 0 through F (15) can be represented in one group of four bits.

Convert the Binary Coded Hexadecimal number 1 0 1 1 0 0 0 1 to a Hexadecimal number.
• • • • • • • •

108.

ANS: B1

Solution:

1 0 1 1

B

0 0 0 1

1

Convert the Binary Coded Hexadecimal number 0 0 1 1 1 1 0 0 to a Hexadecimal number.

11

ANS:

3C

Solution:

 $\frac{0}{} \frac{0}{} \frac{1}{} \frac{1}{}$

3

 $\frac{1}{} \frac{1}{} \frac{0}{} \frac{0}{}$ C

Convert Hexadecimal number 3D to Binary Coded Hexadecimal.

100

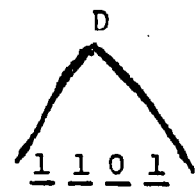
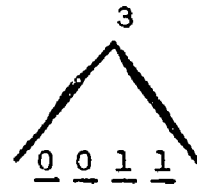
110

ANS:

0 0 1 1

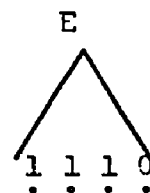
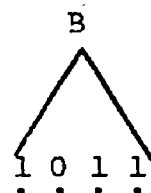
1 1 0 1

Solution:



Convert Hexadecimal number BE to Binary Coded Hexadecimal.

ANS: 1 0 1 1 1 1 1 0 Solution:



TEST

1. Convert Binary Coded Hexadecimal to Hexadecimal.

	<u>8 4 2 1</u>	<u>8 4 2 1</u>	<u>8 4 2 1</u>	<u>256 16 1</u>
a.			0 1 1 1
b.		0 1 1 0	1 1 0 1	6 .
c.		1 1 0 0	0 1 1 0
d.	0 0 1 0	1 0 1 1	1 1 1 1

2. Convert Hexadecimal to Binary Coded Hexadecimal.

	<u>256 16 1</u>	<u>8 4 2 1</u>	<u>8 4 2 1</u>	<u>8 4 2 1</u>
a.			
b.	E 4	
c.	9 A	
d.	2 B B

HEXADECIMAL ADDITION:

$$\begin{array}{r} A4 \\ 6E \\ \hline \end{array}$$

1. Add one column at a time.

First Column:

A	4		4
6	E		+ 14
			18

2. If result is F (15) or less indicate result in column (in Hexadecimal). But if result is more than F, subtract result by 16 (the base of the Hexadecimal number system).

First Column:

A	4		4
6	E		+ 14
			18
			- 16
			2

3. Indicate remainder in the column added and carry 1 to next column:

First Column:

1 carry

1	
A	4
6	E
<hr/>	
	2

4
+14
<hr/>
18
-16
<hr/>
2

Arrows indicate the carry of 1 from the remainder 2 to the next column.

4. Add following columns in same manner (include 1 carry).

Second Column:

1 carry

1	
A	4
6	E
<hr/>	
1	2

1
10
6
<hr/>
17
-16
<hr/>
1

Arrows indicate the carry of 1 from the remainder 1 to the next column.

Third Column:

1	1	
A	A	4
6	6	E
<hr/>		
1	1	2

Thus, the result of the Hexadecimal addition is HEXADECIMAL 112.

Perform Hexadecimal addition:

$$\begin{array}{r} 8 \\ + 3 \\ \hline \end{array}$$

124

114

ANS: B (11)

Perform Hexadecimal addition:

$$\begin{array}{r} B7 \\ 13 \\ \hline 125 \end{array}$$

ANS: C A

Perform Hexadecimal addition:

$$\begin{array}{r} \text{C D} \\ + \text{B E} \\ \hline \end{array}$$

TEST

Perform Hexadecimal addition:

a.
$$\begin{array}{r} 8 \\ + 5 \\ \hline \end{array}$$

b.
$$\begin{array}{r} A6 \\ + 45 \\ \hline \end{array}$$

c.
$$\begin{array}{r} B4 \\ + 6E \\ \hline \end{array}$$

d.
$$\begin{array}{r} DE \\ + CF \\ \hline \end{array}$$

HEXADECIMAL SUBTRACTION

$$\begin{array}{r} A5 \\ + 6D \\ \hline \end{array}$$

1. Subtract one column at a time.

$$\begin{array}{r} \boxed{A} \boxed{5} \\ \hline \boxed{6} \boxed{D} \end{array} \quad \begin{array}{r} \downarrow \\ 5 \\ -1 \ 3 \\ \hline \end{array}$$

2. If minuend (number to be subtracted from) is smaller than subtrahend, borrow the base (16) from next column.

$$\begin{array}{r} 9 \quad \boxed{A} \boxed{5} \\ \hline \boxed{6} \boxed{D} \end{array} \quad \begin{array}{r} \downarrow \\ 2 \ 1 \\ -1 \ 3 \\ \hline \end{array} \quad [5 + 16 \text{ (BASE)}]$$

3. Indicate result in the column subtracted.

First Column:

$$\begin{array}{r} 9 \\ A\ 5 \\ 6\ D \\ \hline 8 \end{array} \qquad \begin{array}{r} 2\ 1 \\ -\ 1\ 3 \\ \hline 8 \end{array}$$

←

4. Subtract following column in same manner.

Second Column:

$$\begin{array}{r} 9 \\ A\ 5 \\ 6\ D \\ \hline 3\ 8 \end{array} \qquad \begin{array}{r} 9 \\ -\ 6 \\ \hline 3 \end{array}$$

←

Thus, the result of Hexadecimal subtraction is 38.

Perform Hexadecimal subtraction:

$$\begin{array}{r} 9\ 4 \\ -\ 7\ C \\ \hline \end{array}$$

ANS: 18

Perform Hexadecimal subtraction

$$\begin{array}{r} C3 \\ - 6B \\ \hline \end{array}$$

TEST

Perform Hexadecimal subtraction:

$$\begin{array}{r} 41 \\ \text{a. } -2B \\ \hline \end{array}$$

$$\begin{array}{r} A2 \\ \text{b. } -87 \\ \hline \end{array}$$

$$\begin{array}{r} C4 \\ \text{c. } -A1 \\ \hline \end{array}$$

$$\begin{array}{r} DC \\ \text{d. } -AC \\ \hline \end{array}$$

DATA REPRESENTATION

PART IV

BCDIC and EBCDIC *

* An understanding of the Hollerith Code is a prerequisite for this part of Programmed Instruction Text. P.I. TEXT on Punched Cards is TC 14-71-50PT.

READ THE FOLLOWING INSTRUCTIONS CAREFULLY BEFORE YOU START THE LESSON

MATERIALS REQUIRED:

In addition to the booklet you should have a pencil and some paper.

A quiet environment that is conducive to strong concentration.

PROCEDURE THAT SHOULD BE FOLLOWED:

Read the frame very carefully.

In some cases the frame should be read twice.

Most of the frames require a response on the part of the student.

Write down the answer that you feel is correct before you turn the page to the answer.

The answer will always be found at the top of the next page.

If your answer is correct continue to the next frame.

If your answer was not correct return to the frame and reread inserting the correct answer as you read.

IF A FRAME MAKES NO SENSE AT ALL TO YOU RAISE YOUR HAND AND WAIT FOR INSTRUCTOR.

INTRODUCTION

To converse with someone who speaks only a foreign language would require one of us to learn another language so we can communicate. Similarly, to communicate with a computer system it is necessary to learn and use it's specific language.

OBJECTIVE

This Programmed Instruction Text covers the various codes used to represent alphabetic, special, and numeric data internally within a computer system. Upon completion of the text you will be able to:

1. Interpret the Binary Coded Decimal Interchange Code.
2. Interpret the Extended Binary Coded Decimal Interchange Code.

BINARY CODED DECIMAL INTERCHANGE CODE

In Binary Coded Decimal, four bits were used to represent one decimal digit and only digits could be represented in this code. By extending this four bit field by two more bits we can represent alphabetic characters as well as numbers. This second generation six bit field is known as the BINARY CODED DECIMAL INTERCHANGE CODE (BCDIC).

B A 8 4 2 1
• • • • •

Name the two punches required on a punched card to represent characters.

ANS: Zone punch and digit punch

The zone and digit bits of the Binary Coded Decimal Interchange Code:

Zone Bits	Digit Bits
<u>BA</u>	<u>8421</u>

Zone bits used to represent zone punches.

<u>(Punch)</u> <u>Zone</u>		<u>(Bits)</u> <u>BA</u>
12	=	11
11	=	10
0	=	01

What letters can be represented in the Hollerith Code with:

- a. 12 zone punch?
- b. 11 zone punch?
- c. 0 zone punch?

126

ANS:

- a. A through I
- b. J through R
- c. S through Z

Representation of characters:

<u>CHARACTER</u>		<u>ZONE PUNCH</u>	<u>DIGIT PUNCH</u>	<u>ZONE BITS BA</u>	<u>DIGIT BITS 8421</u>
A	=	12,	1	11	0001
J	=	11,	1	10	0001
S	=	0,	2	01	0010
9	=	(not used),	9	00	1001

What are the Hollerith Code punches used to represent the letter "B"?

137

ANS: "12" zone punch, "2" digit punch.

Indicate the letter "B" in the Binary Coded Decimal Interchange Code.

128

ANS:

B	A	8	4	2	1
1	1	0	0	1	0

Indicate the letter "M" in BCDIC.

135

ANS:

B	A	8	4	2	1
1	0	0	1	0	0
.

Indicate the letter "V" in BCDIC.

130

ANS:

B	A	8	4	2	1
0	1	0	1	0	1

Indicate the number "6" in BCDIC.

141

ANS:

B A 8 4 2 1
0 0 0 1 1 0

TEST

Indicate character in BCDIC.

	<u>CHARACTER</u>	<u>BA 8 4 2 1</u>
a.	U
b.	D
c.	L
d.	4

Special characters used BCDIC will not be discussed in the Programmed Instruction Text. IBM System 360 Reference Data Card provides the bit configuration of special characters in BCDIC.

PARITY CHECK

Whenever data is transferred from one device to another in a computer system, there is always a chance that a bit condition ("on" or "off") may be lost or gained. This of course, would change the character representation. An additional check bit position has been added to the six bit field (BCDIC) to make what is called a parity check so we can determine when a bit has been lost or gained:

↖ check bit
 C B A 8 4 2 1

When making an odd parity check, the number of "1" bits under the BA8421 are counted. If the number of "1" bits is even, another "1" bit is placed under the check bit (C) to make the entire number of 7 bits odd. For even parity the entire number of 7 bits is even.

The letter "C" with

- | | |
|----------------------------|-------------------|
| 1. <u>Odd</u> Parity: | C B A 8 4 2 1 |
| | 1 1 1 0 0 1 1 |
|
2. <u>Even</u> Parity: |
C B A 8 4 2 1 |
| |
0 1 1 0 0 1 1 |

Place the proper check bit condition ("1" or "0") for the letter "B".
Check for odd parity:

C B A 8 4 2 1
 - 1 1 0 0 1 0
 -

134

ANS:

C	B	A	8	4	2	1
0	1	1	0	0	1	0
—

Place the proper check bit condition for the letter "E". Check for odd parity.

C	B	A	8	4	2	1
—	1	1	0	1	0	1
.

145

ANS:

C	B	A	8	4	2	1
1	1	1	0	1	0	1
-

Place the proper check bit condition for the letter "A". Check for even parity.

C	B	A	8	4	2	1
-	1	1	0	0	0	1
.

136

ANS:

C B A 8 4 2 1
1 1 1 0 0 0 1
- . .

Place the proper check bit condition for the letter "K". Check for even parity.

C B A 8 4 2 1
1 0 0 0 1 0
-

147

ANS: C B A 8 4 2 1
 0 1 0 0 0 1 0

TEST

Place proper check bit condition:

1. Odd Parity

C B A 8 4 2 1

a. - 1 1 0 1 0 0

b. - 1 0 0 1 1 1

c. - 1 1 1 0 0 1

d. - 0 1 0 0 1 1

2. Even Parity

C B A 8 4 2 1

a. - 1 0 0 1 0 1

b. - 0 1 0 1 1 1

c. - 1 1 1 0 0 0

d. - 1 0 0 1 0 0

EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE

In the Binary Coded Decimal Interchange Code, alphabetic characters could be represented as either capital letters (upper case) or small letters (lower case), but not in both cases. By extending the bit, we can represent both upper and lower cases of letters. The new third generation eight bit code is known as the Extended Binary Coded Decimal Interchange Code (EBCDIC).

<u>CASE BITS</u>	<u>ZONE BITS</u>	<u>DIGIT BITS</u>
84 ..	21 ..	8421

8 4

Case bits 1 1 indicate capital letters (upper case).

Case bits 1 0 indicate small letters (lower case).

Hexadecimal numbers are indicated by using 1's for both case & zone bits.

Representation of Characters:

HOLLERITH CODE

CHARACTER		ZONE PUNCH	DIGIT PUNCH	CASE BITS <u>84</u> 11	ZONE BITS <u>21</u> 00	DIGIT BITS <u>8421</u> 0001
A	=	12,	1			
a	=	(not applicable)		10	00	0001
J	=	11,	1	11	01	0001
j	=	(not applicable)		10	01	0001
S	=	0	2	11	10	0010
s	=	(not applicable)		10	10	0010
9	=	(not used),	9	11	11	1001

Note that the "on" and "off" condition of the zone bits in EBCDIC is the reverse of the "on" and "off" condition of the zone bits in BCDIC.

Indicate the following in EBCDIC.

- a. Capital "B"
- b. Small "b"

140.

ANS:

	8421	8421		
a.	1100	0010	=	B
b.	1000	0010	=	b

Indicate the following in EBCDIC.
a. Capital "M"
b. Small "m"

ANS:

a. 8421 8421 = "M"
1101 0100
b. 1001 0100 = "m"

141

Indicate following in EBCDIC.

- a. Capital "V"
- b. Number "7"

142

8421

8421

ANS: a. 1110 0101 = "V" *

 b. 1111 0111 = "7"

*Remember that "V" is 0 zone punch, 5 digit punch.

TEST

Indicate character in EBCDIC.

CHARACTER	8421	8421
a. 4
b. D
c. t (small)
d. 4

153

Special characters used in EBCDIC will not be discussed in this Programmed Instruction. IBM System 360 Reference Data Card provides the bit configuration of special characters in EBCDIC.

A parity check can also be made in EBCDIC by providing an additional check bit. With the exception that there are more bits in EBCDIC than in BCDIC, the method of checking for parity is the same.

The smallest unit of data in a computer is a Bit. Eight bits form the next smallest unit of data, a Byte. In EBCDIC, you've seen that we can use one byte to represent either an alphabetic character, one hexadecimal number, or one special character.

Examples:

BYTE
ZONED DIGIT

1111	1001
------	------

 = 9
F 9

BYTE
ZONE DIGIT

1100	0001
------	------

 = A
C 1

BYTE
ZONE DIGIT

0101	1011
------	------

 = \$
5 B

For representing only numeric data, EBCDIC is redundant and wasteful of storage space in the computer. This is because for numeric data in EBCDIC, the zone half of each byte is a hexadecimal F (1111). The major exception to this is the usage of plus (+) and minus (-) signs, which are shown in the zone portion of the right most byte of an EBCDIC number. To show a positive number the F is changed to a C, and to show a negative number the F is changed to a D.

Examples:

F7	F1	F6	F9
----	----	----	----

 = 7169 (This is assumed positive, although not shown)

F7	F1	F6	C9
----	----	----	----

 = +7169

F7	F1	F6	D9
----	----	----	----

 = -7169

Convert the following numbers to EBCDIC format.

a. 943

c. + 349

b. - 943

d. - 8767

ANS:

a.

F	9	F	4	F	3
---	---	---	---	---	---

c.

F	3	F	4	C	9
---	---	---	---	---	---

b.

F	9	F	4	D	3
---	---	---	---	---	---

d.

F	8	F	7	F	6	D	7
---	---	---	---	---	---	---	---

As you can see, the F zone is repetitive and wastes storage space. To eliminate this waste, the computer provides the "packed Decimal" or "Binary Coded Decimal" representation. Once again "C" indicates a POSITIVE (+) number, and the "D", a NEGATIVE (-) number. The difference is that each byte contains 2 decimal digits, with the sign shown in the "DIGIT" portion of the rightmost byte in the number.

Example:

Decimal number + 63264

Represented in:

EBCDIC

F	6	F	3	F	2	F	6	C	4
---	---	---	---	---	---	---	---	---	---

Packed Decimal

6	3	2	6	4	C
---	---	---	---	---	---

Note that instead of using 5 bytes, we are now able to use only 3 bytes, thus a saving of 2 bytes.

If the number you are packing does not completely fill the area you've allocated for it, zero's (0) are used as fill on the left.

Example:

F	8	F	3	F	2	D	1
---	---	---	---	---	---	---	---

EBCDIC

-8321

0	8	3	2	1	D
---	---	---	---	---	---

Packed

Indicate the decimal number +421 in EBCDIC format and in packed format.

146

ANS: F 4 F 2 C 1 EBCDIC

 4 2 1 C Packed

In "Packing", the proper method for calculating how many bytes will be necessary to accomodate a number is to first count the digits in the number. If it is an even number, add two (2) to it. If it's odd, add one (1). Then divide this number by two. The result will be the number of bytes needed.

Example:

+459 = 3 Digits (odd)
 1 Add 1
 4 Divide by 2
 2 # of bytes

4 5 9 C

+ 6324 = 4 Digits (even)
 2 Add 2
 6 Divide by 2
 3 # of bytes

7 6 3 2 4 C

How many bytes will be needed to pack the number?

- a. 8762391
- b. 67821467821431

ANS: A. = 4

B. = 8

You have seen now how alphabetic characters, number, and special symbols are represented in EBCDIC. You have further seen how we can save storage space by using the "Packed Decimal" feature in representing numbers. Proper knowledge of this is necessary in gaining a complete understanding of computer systems.

SPECIAL TEXT

ST 18-150

DRAFT

BASIC COBOL PROGRAMING

Prepared By

UNITED STATES ARMY INSTITUTE OF ADMINISTRATION

Fort Benjamin Harrison, Indiana — 46216

TABLE OF CONTENTS

	Page
INTRODUCTION	3
COBOL LANGUAGE CONSTRUCTION	3
COBOL CHARACTER SET	4
CHARACTERS USED IN WORDS	4
PUNCTUATION RULES	5
CHARACTERS USED IN ARITHMETIC EXPRESSIONS	5
MARGINS AND THE COBOL CODING SHEET	5
WORD	6
COBOL WORDS	8
THE COBOL LANGUAGE STRUCTURE	9
IDENTIFICATION DIVISION	9
PROGRAM-ID	9
AUTHOR	9
DATE-COMPILED	9
REMARKS	11
ENVIRONMENT DIVISION	12
SPECIAL-NAMES	13
SELECT	17
DATA DIVISION	17
FD	21
RECORD DESCRIPTION	24
WORKING-STORAGE SECTION	24
77 LEVEL	24
USAGE	24
VALUE	24
SYNCHRONIZED	26
JUSTIFIED	26
COMPUTATIONAL	29
EDITED FIELDS	31
ARRAYS	32
REDEFINED	33
88 LEVEL	34
PROCEDURE DIVISION	35
PROCESS VERBS	35
MOVE	40
ADD	42
SUBTRACT	43
MULTIPLY	44
DIVIDE	45
COMPUTE	

	Page
INPUT-OUTPUT	46
READ	46
WRITE	47
DISPLAY	50
EXHIBIT	50
OPEN	51
CLOSE	52
LOGIC FLOW	52
IF	52
PERFORM	59
GO TO	62
STOP	63
ON	64
DEBUGGING	65
TRACE	65
RESET	65
TABLE HANDLING	65
SUBSCRIPTING	66
INDEXING	68
DOS JCL	
OS JCL	

AMERICAN NATIONAL STANDARD COBOL
for
IBM System/360 Disk Operating System & Operating System

This manual is designed as a student introductory text to 'ANSI' COBOL. Since this is an introductory text, information contained herein may contain oversimplified explanations of the COBOL syntax. More explicit information may be obtained from IBM System/360 Disk Operating System American National Standard COBOL, File Number S360-24, GC28-6394 or IBM System/360 Operating System American National Standard COBOL GC28-6396 (IBM Systems Reference Library, DOS). Information contained herein is extracted mostly from these publications. Army standards were taken from U.S. Army Computer Systems Command Manual 18-1-1, dated 15 May 1975.

The following extract from Government Printing Office Form Number 1965-0795689 is presented for the information and guidance of the user:

"Any organization interested in reproducing the COBOL report and specifications in whole or in part, using ideas taken from this report as the basis for an instruction manual or for any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention 'COBOL' in acknowledgement of the source, but need not quote this entire section."

"COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations." "No warranty, expressed or implied, is made by any contributor or by the COBOL Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee in connection therewith." Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages." "The authors and copyright holders of the copyrighted material used herein have specifically authorized the use of this material in whole or

FLOW-MATIC (Trademark of Sperry Rand Corporation), Programming for the UNIVAC (R) I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial translator, Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260, copyrighted 1960 by Minneapolis-Honeywell.

in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications."

Information in this text represents the language promulgated by the Conference on Data Systems Languages (CODASYL) (the USA standard of the language is American National Standard COBOL, X3.23-1968 (formerly known as USA Standard COBOL, as approved by the American National Standards Institute (ANSI), unless

DRAFT ST 18-150

specifically stated otherwise. COBOL is a problem oriented language which is designed to be highly self-documenting. The structure of the language helps document the program as it is written.

INTRODUCTION

COBOL LANGUAGE CONSTRUCTION. COBOL is based on English. It uses English-type words and certain syntax rules derived from English. However, because it is a computer language, it is much more precise than English. The programmer must, therefore, learn the rules that govern COBOL and follow them exactly.

The basic unit of COBOL is the word. This may be either a COBOL reserved word or a programmer-defined word. Reserved words have a specific syntactical meaning to the COBOL compiler and must be spelled exactly as shown. Programmer-defined words are assigned by the user to such items as data-names (a name made up by the programmer to refer to a specific piece of data in memory). Reserved words and programmer-defined words are combined by the programmer into clauses and statements.

Clauses and statements must be formed following the specific syntactical rules of COBOL. A clause or a statement specifies only one action to be performed, one condition to be analyzed, or one description of data. Clauses and statements can be combined into sentences.

Sentences may be simple (one statement or combination of clauses). Sentences can be combined into paragraphs, which are named units of logically related sentences, and paragraphs can be further combined into named sections.

Both paragraphs and sections can be referred to as procedures, and their names can be referred to as procedure names.

Procedures (sections and paragraphs) are combined into divisions. Divisions are joined into a COBOL program. Each program has exactly four divisions.

COBOL CHARACTER SET. The following symbols may be used in writing COBOL:

<u>SYMBOL</u>	<u>MEANING</u>	<u>SYMBOL</u>	<u>MEANING</u>
0,1,...9	digits	,	comma
A,B,...Z	letters	;	semicolon
+	plus sign	.	period
-	minus/hyphen sign	'	quotation mark (ANSI shows a quote as ")
*	asterisk	(left parenthesis
/	slash)	right parenthesis
=	equal sign	>	greater than symbol
\$	currency sign	<	less than symbol
			space (no printed character)

Characters Used in Words

The characters used in words in a COBOL source program are:

0 thru 9
A thru Z
- (hyphen)

A word is composed of a combination of not more than 30 characters chosen from the character set for words. The word cannot begin or end with a hyphen.

The following characters are used for punctuation:

space	.	period
,)	right parenthesis
;	(left parenthesis
'		quotation mark

PUNCTUATION RULES

The following general rules of punctuation apply in writing COBOL source programs:

1. A period, semicolon, or comma, when used, must not be preceded by a space, but must be followed by a space.
2. A left parenthesis must not be followed immediately by a space; a right parenthesis must not be preceded by a space.
3. At least one space must appear between two successive words and/or parenthetical expressions and/or literals. Two or more successive spaces are treated as a single space, except within non-numeric literals.
4. An arithmetic operator or an equal sign must always be preceded by a space and followed by a space.
5. A comma may be used as a separator between successive operands of a statement. An operand of a statement is shown in a format as a lower case word.
6. A comma or semicolon may be used to separate a series of clauses and is used for ease of reading as in written English.
7. A semicolon may be used to separate a series of statements. Example:
ADD A TO B; SUBTRACT B FROM C.

Characters Used in Arithmetic Expressions:

+ addition
 - subtraction
 * multiplication
 / division
 ** exponentiation
 (left parenthesis
) right parenthesis
 = replacement operator

NOTE: When any punctuation mark is indicated in a format (in this publication), it is required in the program.

MARGINS AND THE COBOL CODING SHEET

When writing a COBOL program, the programmer is concerned with 2 margins: margin A and margin B. Margin A encompasses columns 8 through 11 of a coding sheet. Margin B encompasses columns 12 through 72.

Margin A is used to identify the different divisions, paragraphs, sections, or special items of a COBOL program. Entries in Margin A may continue through column 72, if necessary, and are terminated by a period. Each statement or clause should be on a single line (card). Margin A statements do not have to begin in column 8, but must begin before column 12.

Army Standard: All entries that can begin in Margin A must begin in column 8.

Margin B contains all the information the programmer wishes to include within a defined division, paragraph, section, or special items of the program. Margin B statements do not have to begin in column 12, but may be written anywhere from column 12 through column 72 (see below).

EXAMPLE:

SPECIAL-NAMES.

CØ1 IS CHAN 1.

The word SPECIAL-NAMES begins in margin A.
 The words CØ1 IS CHAN 1 begins in margin B.

WORD

A WORD is composed of not more than 30 characters from the COBOL character set for word formation. The space (blank) is not an allowable character in a word; the space is a word separator. A word is terminated by a space, a period, a right parenthesis, a comma, or a semicolon.

Throughout this publication, basic formats are prescribed for various elements of COBOL. These generalized descriptions are intended to guide the programmer in writing his own statements. They are presented in a uniform system of notation, explained in the following paragraphs. Although it is not part of COBOL, this notation is useful in describing COBOL.

COBOL WORDS

1. All words printed entirely in capital letters are reserved words. These words have preassigned meanings in COBOL. In all formats, words in capital letters represent an actual occurrence of those words. If any such word is incorrectly spelled, it will not be recognized as a reserved word and will probably cause an error in the program. Reserved words cannot be used as procedure or data names.
2. All underlined reserved words are required unless the portion of the format containing them is itself optional. These are key words. If any such word is missing or is incorrectly spelled, it is considered an error in the program. Reserved words not underlined may be included or omitted at the option of the programmer. These words are used only for the sake of readability; they are called optional words, and when used, must be correctly spelled.
3. The characters + - = when appearing in formats, although not underlined, are required when such formats are used.
4. All punctuation and other special characters (except those symbols cited in the following paragraphs) represent the actual occurrence of those characters. Punctuation is essential where it is shown. Additional punctuation can be inserted according to the rules for punctuation specified in this publication.
5. Words that are printed in lower-case letters represent information to be supplied by the programmer. No entry in lower case letters may be omitted, unless it is part of an optional section or phrase which is omitted.
6. In order to facilitate references to them in text, some lower-case words are followed by a hyphen and a digit or letter. This modification does not change the syntactical definition of the word.
7. Square brackets ([]) are used to indicate that the enclosed item may be used or omitted depending on the requirements of the particular program. When two or more items are stacked within brackets, one or none of them may occur.
8. Braces ({}) enclosing vertically stacked items indicate that one of the enclosed items is obligatory.

9. The ellipsis (...) indicates that the immediately preceding unit may occur once, or any number of times in succession. A unit means either a single lower-case word, or a group of lower-case words and one or more reserved words enclosed in brackets or braces. If a term is enclosed in brackets or braces, the entire unit of which it is a part must be repeated when repetition is specified.
10. Programmer defined words are names chosen by the programmer to identify program-unique things such as file names or data names. They should be descriptive and clear to someone not familiar with the program.

Army Standard: Only Army standard abbreviations in AR 18-12 may be used in programmer defined words.

THE COBOL LANGUAGE STRUCTURE

There are four divisions in each COBOL program. Each is placed in its logical sequence, each has its necessary logical function in the program, and each uses information developed in the divisions preceding it. The four divisions and their sequences are:

IDENTIFICATION DIVISION.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.

IDENTIFICATION DIVISION

The purpose of the IDENTIFICATION DIVISION is to give the who-what-when-why-where-how of the program to a maintenance programmer. It also identifies the program to the operating system.

BASIC FORMAT:

IDENTIFICATION DIVISION.

<u>PROGRAM-ID.</u>	program-name.
<u>AUTHOR.</u>	your name.]
<u>INSTALLATION.</u>	where the program is written.]
<u>DATE-WRITTEN.</u>	the date that coding began.]
<u>DATE-COMPILED.</u>	this entry is completed by the compiler.]
<u>SECURITY.</u>	the classification of the program.]
<u>REMARKS.</u>	a brief description of what the program accomplishes.]

The first words of any COBOL program are IDENTIFICATION DIVISION.
IDENTIFICATION DIVISION must begin in margin A:

IDENTIFICATION DIVISION.

Note that there is one space between IDENTIFICATION and DIVISION, and that IDENTIFICATION DIVISION terminates with a period.

The only required part of the IDENTIFICATION DIVISION is the PROGRAM-ID clause. Since "program-name" is in lower case letters, the programmer must make up a descriptive name by which his program may be identified. All other IDENTIFICATION DIVISION entries help to describe and document your program.

IDENTIFICATION DIVISION.

PROGRAM-ID. SAMPLE-1.

Note that PROGRAM-ID is written in margin A and follows IDENTIFICATION DIVISION which is written in margin A. After PROGRAM-ID, there is at least one space (a period must be followed by one or more spaces). Then the programmer writes the name of his program, in this case, SAMPLE-1. The program name may be placed anywhere in margin B. An alternate example, which means exactly the same thing as the first example, is listed below.

EXAMPLE:

IDENTIFICATION DIVISION.

PROGRAM-ID. SAMPLE-1.

All other clauses of the IDENTIFICATION DIVISION are optional. They may be included or deleted as desired. Each clause begins with the word AUTHOR, INSTALLATION, DATE-WRITTEN, DATE-COMPILED, SECURITY, or REMARKS. Each is followed by a period, followed by any comment-type entry, then terminated by a period. This entry, called a sentence, may be repeated as many times as desired.

EXAMPLE:

IDENTIFICATION DIVISION.

PROGRAM-ID. PILPMGT

AUTHOR. LT HUDGIN, ATSG-D-S:

INSTALLATION. USAIA, FORT HARRISON, IN 46216.

DATE-WRITTEN. 14 AUGUST 1976.

DATE-COMPILED. TODAY

SECURITY. UNCLASSIFIED.

*REMARKS. THIS IS A SAMPLE OF A COMPLETE IDENTIFICATION
 * DIVISION. ALL OF THE ENTRIES ARE INCLUDED IN THIS EXAMPLE,
 * ALTHOUGH ONLY THE PROGRAM-ID PARAGRAPH (WHICH INCLUDES THE
 * PROGRAM-NAME) IS REQUIRED.

Army Standards: Although most entries are not required by ANS COBOL, it is a good practice to include them for documentation purposes.

PROGRAM-ID: The PROGRAM-ID will consist of six positions constructed as follows:

Position	Entry
1	a. For a program that is operational, this position will contain a 'P'. b. For a program that is being tested, this position will have a 'T'. c. If the program is under development, enter a 'D' in this position.
2-3	These two positions identify the specific program within a system or subsystem.
4-6	The subsystem's or system's identification code.

AUTHOR: Enter the programmer's name and/or the office symbol responsible for the program.

DATE-COMPILED: Some compilers will not properly place the compilation date if entry is blank. To avoid this, enter 'TODAY' in this area.

REMARKS: Some compilers do not support this paragraph. To avoid errors, enter an asterisk (*) in column 7 of every card in this paragraph. This paragraph should give the reason the program was written, basically how it works, any special processing or techniques used, what all 77 and 01 entries in the WORKING-STORAGE SECTION are used for, and any communications with other programs. Each time a change is made, the maintenance programmer should add an entry giving the date, his name, why the change was made, and what the change encompassed.

ENVIRONMENT DIVISION

The ENVIRONMENT DIVISION is used for the following purposes:

- a. To identify the computer system to be used (e.g., IBM 360-30, UNIVAC 1108, etc.).
- b. To identify the peripheral devices to be used (e.g., card readers, printers, tape drives, etc.).
- c. To give or assign a logical name to each peripheral device (e.g., SYS006, SYS004, etc.).
- d. To identify the files to be used by the program.
- e. To define the organization (sequential, random, etc.) of each file.
- f. To assign each file to a specific peripheral device.

BASIC FORMAT:

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

[SOURCE-COMPUTER. computer-name.]

[OBJECT-COMPUTER. computer-name.]

[SPECIAL-NAMES. [function-name-1 IS condition-name-1]]...

INPUT-OUTPUT SECTION.

FILE-CONTROL.

(DOS)*

$$\begin{array}{l} \text{SELECT file-name} \\ \text{ASSIGN TO} \end{array} \quad \text{SYSnnn-} \left\{ \begin{array}{c} \text{UT} \\ \text{UR} \end{array} \right\} = \left\{ \begin{array}{c} 2314 \\ 1403 \\ 2400 \\ 2540R \\ 2540P \end{array} \right\} = \underline{S}.$$

(OS)*

$$\begin{array}{l} \text{SELECT file-name} \\ \text{ASSIGN TO} \end{array} \quad \left\{ \begin{array}{c} \text{UT} \\ \text{UR} \end{array} \right\} = \left[\begin{array}{c} 2314 \\ 1403 \\ 2400 \\ 2540R \\ 2540P \end{array} \right] = \underline{S} = \text{external-name.}$$

* See note on next page.

* This paragraph by definition is machine and operating system dependent. We have provided examples of its coding for the two most common operating systems in the Army. No attempt is made to limit this instruction only to IBM hardware or software. Each manufacturer has programming manuals available to explain their own implementation of this paragraph. If device independence is desired when using OS, the device model number must be omitted from the system name and the device class must be UT even if the file is on a unit record device when the program is executed.

There are 2 sections of the ENVIRONMENT DIVISION: The CONFIGURATION SECTION and the INPUT-OUTPUT SECTION. The CONFIGURATION SECTION specifies which computer is to be used to compile the program (the SOURCE-COMPUTER), which computer will be used to execute the program once it is compiled (the OBJECT-COMPUTER), and associates names in the COBOL program with special tests on peripheral devices used in the program.

EXAMPLE:

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-360-G30.
OBJECT-COMPUTER. IBM-360-G30.

In this example, both the source and object computers are the same. The descriptions should be of the smallest computer that the program can be compiled or executed on.

SPECIAL-NAMES is part of the CONFIGURATION SECTION and is used if the programmer wishes to skip to a particular line on the printer page (for example, to the top of the next page). The IBM 1403 printer has a carriage control tape with 12 channels, each of which may have a hole punched in it to indicate where to stop the paper if that particular channel is searched. The carriage control tape used for student jobs has a punch indicating the top of page is on channel 1 and the bottom of the page is on channel 09. The programmer must specify a name which will be associated with the hole punched in the channel if he wishes to skip (slew) to that point on the page. The names to be associated with the channels are specified in the SPECIAL-NAMES clause:

Army Standard: Both the SOURCE-COMPUTER and OBJECT-COMPUTER are required entries.

SPECIAL-NAMES.
Cnn IS mnemonic-name(...).

"C" means channel; nn is a two-digit number indicating which channel of the twelve possible is to be associated with the name. The mnemonic name is the name that the programmer gives to the channel. It will be used in the PROCEDURE DIVISION skipping to the channel is desired.

EXAMPLE:

SPECIAL-NAMES.

CØ1 IS CHAN1

CØ9 IS CHAN9.

Here the word CHAN1 will later be recognized as meaning skip to channel 1 of the carriage control tape on the printer. CHAN9 will mean skip to channel 09.

Army Standards: The only special names that may be used and their purpose are:

Special Name	Purpose
CHAN1	Indicates the first print line page (Channel 1).
CHAN9	Indicates the last print line of a page (Channel 9).

Channel 1 on the Army standard channel tape is line 6 on the printer page. Channel 9 is line 66. On standard 11" long printer paper printed six lines per inch that leaves one inch margins at the top and bottom.

The FILE-CONTROL paragraph of the INPUT-OUTPUT SECTION is used to tell the system how many files will be used, what type of peripheral devices the files are on, the names to be used to refer to each file, the organization of the files, and where the files are located. Each file used is specified at least once by the use of a SELECT clause and its parts.

To tell the system that a file will be used, write the word SELECT (following the INPUT-OUTPUT SECTION and FILE-CONTROL cards) in margin B. Then devise a name for the file. This name is written after SELECT.

EXAMPLE 1:

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT MASTER-FILE

EXAMPLE 2:

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT INVENTORY

In the above 2 examples, the programmer made up the word MASTER-FILE in the first example, and the word INVENTORY in the second. From this point on, these words must be used whenever the programmer wishes to do anything to the file (e.g., READ or OPEN). The name chosen for the file cannot be a COBOL reserved word.

After the file name, the programmer writes ASSIGN TO followed by a system name. The system name can have two forms: one for DOS and one for OS. A DOS system name for a file consists of the logical unit assignment (system logical numbers), the class of device, the model number of the specific device, and the file organization. An OS system name for a file consists of the class, the file organization, and the external name.

In DOS, the system logical number refers to a specific hardware device assigned to that number by JCL (Job Control Language). Its value is a number from SYS000 to SYS243, depending on the parameters chosen during system generation. DOS and OS JCL are discussed later in this text. In OS, the external name performs the same function as the system logical number in DOS. It is limited to from 1 to 8 characters, must begin with a letter, and contain no special characters.

Class refers to the device's capability to handle one or more than one record length. A punched card, for example, is always the same length; it has 80 characters represented on it at all times. UNIT-RECORD (UR) is the class used to describe this type device. Even though no punches are punched in the card, there are still 80 characters there; the space is a character to the computer. When read, therefore, there must be sufficient memory allocated to hold 80 characters, whether they are spaces or any of the other permissible characters of the HOLLERITH or EBCDIC code. The same is true when punching a card, the card punch must know what character is to be punched in each of the 80 columns. A blank card to be punched means, in effect, punch no holes. A similar concept holds true for the printer which must know what character is to be printed in each of 132 print positions. These devices, the card reader, card punch, and printer, are known as UNIT-RECORD devices. They always handle fixed-length records. On the other hand, magnetic tape and disk units are not limited to just one record length. These devices conform to the record length specified for each particular job or file, and when accessing information sequentially, they are referred to as UTILITY (UT) devices.

The letters UR are used to specify UNIT-RECORD; the letters UT are used to specify UTILITY.

After the class is specified, the programmer specifies the model number of the device to be used for the file. This is the hardware model number of the peripherals in the machine configuration at the installation.

For an IBM 360-30 or 360-40, which are the most common Army configurations,

The card reader is specified by 2540R.

The card punch is specified by 2540P.

The disk is specified by 2314.

The printer is specified by 1403.

The magnetic tape is specified by 2400.

The file organization will always be sequential for UR and UT devices.

Sequential organization is specified by S.

All of the parts in the system name are separated by hyphens.

EXAMPLE-1 (DOS). A card file (input):

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT MASTER-INVENTORY-FILE
ASSIGN TO SYS006-UR-2540R-S.

EXAMPLE-1 (OS).

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT MASTER-INVENTORY-FILE
ASSIGN TO UT-S-MASTER.

The DOS example shows that the name of this file is MASTER-INVENTORY-FILE. The system logical number is SYS006; the card reader is a unit-record device, thus UR; the device number for the card reader is 2540R; and S indicates that it is a SEQUENTIAL file.

The OS example has the same file name, but the device model number is omitted, the class is UT, the organization is sequential, and the external name is MASTER.

EXAMPLE-2 (DOS). An output file on a printer:

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT LISTING
ASSIGN TO SYS005-UR-1403-S.

EXAMPLE-2 (OS).

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT LISTING
ASSIGN TO UT-S-PRINTER.

The DOS example shows that the name of the file is LISTING; the system logical number is SYS005; UR for UNIT-RECORD; 1403 which is the device number for the printer; and S for SEQUENTIAL file.

The OS example has the same file name, but the device model number is omitted, the class is UT, the organization is sequential, and the external name is PRINTER.

If a program required 2 files, one for input and the other for output, 2 select statements would be required.

EXAMPLE-3 (DOS). A program requiring 2 files--input from disk and output on a printer:

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT INPUT-FILE

ASSIGN TO SYS006-UT-2314-S.

SELECT OUTPUT-FILE

ASSIGN TO SYS005-UR-1403-S.

EXAMPLE-3 (OS).

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT INPUT-FILE

ASSIGN TO UT-S-DISK FILE

SELECT OUTPUT-FILE

ASSIGN TO UT-S-PRINTER.

Note that there is one select statement for each file in the program. The outputs file are similar to the ones in Example 2. For the input file:

DOS

The name of the file is INPUT-FILE.
The system logical number is SYS006.
UT for a UTILITY device.
2314 which specifies the type of
disk drive on our system.
S for SEQUENTIAL file.

For the output file:

DOS

The name of the file is OUTPUT-FILE.
The system logical name is SYS005.
It is a UTILITY device (UT).
The device is an IBM 1403 printer.
It is a sequential file (S).

OS

The name of the file is INPUT-FILE.
The device class is UT.
The organization is sequential.
The external name is DISK FILE.

OS

The name of the file is OUTPUT-FILE.
The device class is UT.
The organization is sequential.
The external name is PRINTER.

DATA DIVISION

The purpose of the DATA DIVISION is:

1. To describe each file in detail.
2. Allocate memory required by the record of each file.
3. Allocate all other memory needed by the program for working areas (scratch pad).

This text explains the DATA DIVISION in 3 parts: the FILE SECTION FD (file description), the FILE SECTION record description, and the WORKING-STORAGE SECTION.

DATA DIVISION.

FILE SECTION.

FD file-name

[BLOCK CONTAINS integer [CHARACTERS
RECORDS]]

[RECORD CONTAINS integer-1 [TO integer-2 CHARACTERS]]

LABEL {RECORD IS
RECORDS ARE} {OMITTED
STANDARD}

[DATA {RECORD IS
RECORDS ARE} data-name-1 [data-name-2] ...]

Army Standard: When more than one clause is used to describe a file, each clause must begin on a new card beginning in column 12.

The first part of the DATA DIVISION which is described is the File Description. FD in margin A identifies that what follows is a description of one of the files identified by the SELECT clause in the ENVIRONMENT DIVISION. After FD, place the name of the file to be described in margin B, followed by one or more of the clauses shown above, as applicable. The name of the file must be identical to that of the SELECT clause.

The only required clause is the LABEL clause. All others are optional. It is recommended that the other clauses are used to increase the efficiency of the compiler and for documentation.

All disk files have standard labels. All unit record files have omitted labels.

Army Standard: All possible entries are given for all file descriptions. All tape files must have standard labels unless created on another installation and used as input.

Under OS with no device model number specified in the SELECT statement, the label records are always standard regardless of the device to which the file is ultimately assigned. When used for magnetic tape, labels may appear at the beginning and the end of the file and are used to prevent the wrong program from accessing the information on the tape. When used for disk, labels are used to locate the beginning and end of sequential files. Since the system must know where a sequential file begins and ends on disk, labels must be used on disk; however, they are not mandatory for tape files. It is good programming practice to always use labeled files. This provides another check for the validity of the input or output data.

EXAMPLE (DOS & OS):

FILE SECTION.

FD MASTER-IN

LABEL RECORD IS STANDARD.

In this case, MASTER-IN was the name assigned to a file in a SELECT statement in the ENVIRONMENT DIVISION. This file was assigned to either a magnetic tape file or a disk file, if under DOS. All files have standard labels under OS. The LABEL RECORD IS STANDARD, which means to the system, generate standard labels according to my JCL instructions for output files or use the existing standard labels for input files.

If the file called MASTER-IN was assigned to a card reader, the FD would appear as:

EXAMPLE (DOS):

FILE SECTION.

FD MASTER-IN

LABEL RECORD IS OMITTED.

EXAMPLE (OS):

FILE SECTION.

FD MASTER-IN

LABEL RECORD IS STANDARD.

Since LABEL RECORDS can be used only for magnetic tape and disk under DOS, they must be OMITTED for card readers, punches, and printers. They may also be OMITTED under DOS for magnetic tape if a non-labeled tape is being used. OS always uses standard labels with device independence.

EXAMPLE (DOS):

FILE SECTION.

FD PRINTOUT

LABEL RECORD IS OMITTED.

EXAMPLE (OS):

FILE SECTION.

```
FD PRINTOUT
   LABEL RECORD IS STANDARD.
```

DOS: In this case, a file called PRINTOUT, assigned to the 1403 printer in the ENVIRONMENT DIVISION, is described in its file description so that LABEL RECORDS ARE OMITTED.

OS: All devices have standard labels in OS regardless of the class.

There are other parts of the FD which may be used to reduce compile time for your program, and for more sophisticated programs, to reduce execution time. (See Appendix C, ANSI COBOL MANUAL.)

The BLOCK CONTAINS clause specifies the blocking factor--that is, how many records are to be read or written at once for each physical input/output operation. The BLOCK CONTAINS must be equal to 1 or omitted for unit-record devices (card readers, card punches, and printers), because a card is physically read, punched or a line is written each time the COBOL program READs or WRITEs a logical record. Physical READs and WRITEs take a lot of time when compared to the internal speed of the computer. So the programmer, when using utility devices, can lump together several logical records into one physical record before it is read or written. This cuts down the number of READs and WRITEs on a device and speeds up the execution of the program. The BLOCK CONTAINS clause is needed in this case to tell the compiler that several logical records will be read or written each time the system performs a physical READ or WRITE to or from a device. As far as the programmer is concerned, nothing changes in the program except the addition of the BLOCK CONTAINS clause indicating how many of the logical records are to be included in the block. On unit record devices, the logical record must be the same size as the physical record. If BLOCK CONTAINS is omitted, the system defaults to a blocking factor of 1.

EXAMPLE:

FILE SECTION.

```
FD TAPE-FILE
   BLOCK CONTAINS 3 RECORDS
   LABEL RECORD IS STANDARD.
```

This entry would not write to the file called TAPE-FILE until 3 records had been built up into a block during processing via WRITE statements. If the file is input, then a block of 3 logical records is read and one is given to the program each time a READ statement for the file is executed.

The RECORDS CONTAINS clause specifies the length of each logical record in the file. A card, for example, contains 80 characters. Thus, the programmer may write:

170

FILE SECTION.
FD TAPE-FILE
RECORD CONTAINS 80 CHARACTERS
LABEL RECORDS OMITTED.

Note that, in the BLOCK CONTAINS clause, the programmer may specify the number of characters instead of the number of records. If, on magnetic tape, each record was 40 characters in length, the FD could be written.

FILE SECTION.
FD TAPE-FILE,
BLOCK CONTAINS 120 CHARACTERS,
LABEL RECORDS OMITTED.

or

FILE SECTION.
FD TAPE-FILE,
BLOCK CONTAINS 3 RECORDS,
LABEL RECORD IS OMITTED.

or

FILE SECTION.
FD TAPE-FILE,
BLOCK CONTAINS 120 CHARACTERS,
RECORD CONTAINS 40 CHARACTERS,
LABEL RECORD IS OMITTED.

NOTE: Commas have no effect on the meaning of anything written. The programmer may use either or both RECORD CONTAINS or BLOCK CONTAINS. Label records may be either STANDARD or OMITTED for tape files, depending on the desires of the programmer. The reason that the RECORD CONTAINS clause may be omitted is that, following the FD clause, COBOL requires that the record for the file be described in detail. The compiler can, therefore, obtain this information from the description of the record (discussed after the FD clause in this text).

The DATA RECORD clause is used to give a name to the record for a file. It is the record which must be placed in memory; therefore, the description of the record is used to allocate storage. To initially specify what the record's name shall be, the programmer uses the DATA RECORD clause.

Army Standard: To avoid duplicate data names, the data names chosen for records descriptions should be prefixed by a four character identifier record and file for that file. It should be in the form AANN where AA is an identifier unique for the file and NN is a two digit number unique for that record in the file. The first character of the file identifier should specify whether the file is input, output, or work ("I", "O", or "W"). The second character should describe the type of file (e.g., "M" for master file, "T" for transactions file, etc.).

181

EXAMPLE-1:

FILE SECTION.

FD INPUT-FILE

LABEL RECORD IS OMITTED

DATA RECORD IS ITØ1-INPUT-RCD.

EXAMPLE-2:

FILE SECTION.

FD MASTER-INVENTORY-FILE

LABEL RECORD IS STANDARD

DATA RECORD IS OMØ1-MASTER-INVENTORY-RCD.

In the first example, the name of the record for INPUT-FILE is ITØ1-INPUT-RCD. The prefix means that it is the first ("Ø1") record description of an input ("I") transaction ("T") file. In the second example, the name of the record for MASTER-INVENTORY-FILE is OMØ1-MASTER-INVENTORY-RCD. The prefix means that it is the first ("Ø1") record description of an output ("O") master ("M") file.

In the entire FD, there is only one period--the one at the end of all entries.

RECORD DESCRIPTIONS

A record is described by placing 'Ø1' in margin A, followed by the name of the record in margin B, followed by a description (a PICTURE) of the record.

EXAMPLE:

FILE SECTION.

FD MASTER-FILE

LABEL RECORD IS STANDARD

DATA RECORD IS IMØ1-MASTER-RCD.

Ø1 IMØ1-MASTER-RCD

PICTURE IS X(1ØØ).

In the above example, IMØ1-MASTER-RCD is the name of the record for MASTER-FILE. The prefix means that it is the first ("Ø1") record description of an input ("I") master ("M") file. Since the label record is STANDARD, this file must either be on a disk or magnetic tape file. The line which begins with Ø1 (which is called a level number is used to allocate storage (memory) for the record called IMØ1-MASTER-RCD. In this case, IMØ1-MASTER-RCD has a length of 1ØØ (there are 1ØØ characters in the record).

PICTURE CLAUSE.

The PICTURE clause is used to allocate memory. There are 4 types of information which a PICTURE clause describes:

an alphanumeric field, designated by X's.
 an alphabetic field, designated by A's.
 a numeric field, designated by 9's and optionally a P, S, and/or V.
 an edited field, designated by a combination of the characters
 Z . , * \$ - + S V B CR and DB.

Army Standard: (1) The PICTURE clause must be punched beginning in column 48.
 (2) PICTURE is preferred abbreviated as PIC.
 (3) When more than one clause is used to describe a data item,
 each clause must begin on a new card beginning in column 40.

An alphanumeric field (X) can contain any legal EBCDIC character. An alphanumeric literal (also called nonnumeric literal) is a character or series of characters enclosed in quotes and is handled as if it were an X (alphanumeric) field. For the continuation of an alphanumeric literal, use all columns through column 72 on the current card, release the card, and punch a dash (-) in column 7 of the next card. The alphanumeric literal is then continued by placing a quote somewhere in margin B and continuing the literal after the quote as though no continuation had been done. It is finished as a normal alphanumeric literal with a quote and period following the end of the alphanumeric literal.

A numeric field (9) is the only field which can be used in mathematical computations or tests and cannot contain any non-numeric characters.

An edit field is designed to edit a numeric or alphanumeric field (e.g., put in dollar signs, decimal points, commas, blanks, etc.)

In the example:

01 OMØ1-MASTER-RCD PIC X(100).

OMØ1-MASTER-RCD is described as an alphanumeric field which can hold 100 characters. These characters can be any legal EBCDIC character, including any letter of the alphabet, zero thru nine, blanks, and any special characters such as commas, periods, asterisks, etc.

In order to refer to specific parts of a whole record, the record is "broken down" into smaller parts.

EXAMPLE: A punched card contains the name of an individual in columns 1-20;
 the individual's age in columns 21-23;
 spaces in columns 24-80.

To describe this card, the programmer breaks the record into parts, using a higher level number to indicate that NAME, for example, is part of CARD-REC. The same holds true for AGE and FILLER. NOTE: Level numbers indicate the hierarchy of data within a record. The range is 01 through 49. Level 01 is highest level and indicates a record in the FILE SECTION. The higher the value of the number, the lower the level indicated. A field of a record may be assigned level number 3. Its subfields may be assigned level number 4.

Level numbers need not be consecutive, but must be sequential. The record level, must always be 01.

Army Standard: Subordinate data items of a record description all will be prefixed by the record prefix. This identifies them uniquely throughout the program. A fifth character may be added between the second and third characters if two files may have the same prefix, e.g., two input master files may be prefixed by IM101 and IM201 respectively.

FILE SECTION.

FD CARD-FILE

LABEL RECORD IS OMITTED

DATA RECORD IS IT01-CARD-RCD.

01 IT01-CARD-RCD.

05 IT01-NAME

PIC X(20).

05 IT01-AGE

PIC 999.

05 FILLER

PIC X(57).

In the above example, IT01-NAME, IT01-AGE, and FILLER are all part of IT01-CARD-RCD, because the level number 05 is lower than 01. All numbers except 01, (which is discussed later) appear in margin B. Note that the number in parenthesis (called a duplication factor) indicates the number of times the preceding character is to be repeated. The above example could be written without the duplication factor:

FILE SECTION.

FD CARD-FILE

LABEL RECORD IS OMITTED

DATA RECORD IS IT01-CARD-RCD.

01 IT01-CARD-RCD.

05 IT01-NAME

PIC XXXXXXXXXXXXXXXXX.

05 IT01-AGE

PIC 999.

05 FILLER

PIC XXXXXXXXXXXXXXXXXXXXXXXXX.

05 FILLER

PIC XXXXXXXXXXXXXXXXXXXXXXXXX.

Notice that the FILLER with a length of 57 characters had to be broken into two pieces. That is, the number of characters used to represent the PICTURE cannot be longer than 30 characters, not the number of characters that PICTURE actually describes.

If a level is to be broken down into smaller parts, it contains no PICTURE clause and is referred to as a group item. The length of a group item is the sum of its subordinate items. To break IT01-NAME down into a first name field and a last name field, the programmer could write:

01 IT01-CARD-RCD.

05 IT01-NAME.

10 IT01-FIRST-NAME

PIC X(10).

10 IT01-LAST-NAME

PIC X(10).

05 IT01-AGE

PIC 9(3).

05 FILLER

PIC X(57).

ITØ1-NAME is now a group item and is still equal to a length of 2Ø characters because its parts' lengths add up to 2Ø. In like manner, ITØ1-CARD-RCD is equal in length of 8Ø because its parts' lengths add up to 8Ø.

The DATA RECORD clause need not be specified because the Ø1 entry is the record name and is implied just by being under its respective FD. It is good programing practice to include it.

Army Standard: As level numbers increase in record descriptions, they will go Ø1, Ø5, 1Ø, etc. Columns for each level to begin in are 8, 12, 14, 16, etc., out to column 24. In no case will any level number be punched to the right of column 24.

WORKING-STORAGE SECTION.

The purpose of the FILE SECTION is to describe the files to be used in the program and to allocate memory required for the records of each of the files. The WORKING-STORAGE SECTION is used to allocate all memory needed for holding any data other than records described in the FILE SECTION. All storage in the FILE SECTION is transient. When a record is read, the information that was contained in the record area of the memory before the READ statement is gone forever. When a record is written, the information that was put in the record area of the memory is gone. WORKING-STORAGE, on the other hand, is permanent storage. Variables in this area only change when the program physically moves a new value into them. WORKING-STORAGE should be used for constants; e.g., headers, footers, holding areas, counters, and anything that needs to be kept around during the execution of the program. WORKING-STORAGE is divided into two areas: the 77's area and the Ø1's area. Each of these will be described in detail later. Because these areas are permanent storage, the compiler will allow you to initialize them to a given value. The record areas in the FILE SECTION cannot be initialized by the compiler since the areas change each time an input or output operation takes place.

Army Standard: All data names in the WORKING-STORAGE SECTION are prefixed by the characters WS-.

EXAMPLE:

WORKING-STORAGE SECTION.

```

77  WS-LINE-COUNT                                PIC S9(4)
                                           USAGE IS COMP
                                           VALUE IS ZEROS
                                           SYNCHRONIZED.

Ø1  WS-HEADING-LINE.
    Ø5  FILLER                                PIC X(64)
                                           VALUE IS SPACES.
    Ø5  FILLER                                PIC X(69)
                                           VALUE IS 'TITLE'.

```

A 77 data item is normally used for counters and hold areas. All 77 items appear immediately after WORKING-STORAGE SECTION and before the first 01 record entry. They are always elementary items and therefore cannot be subdivided with larger numbers like 78. In the example above, a 77 item named WS-LINE-COUNT is defined and initialized to a value of zero. The other area in WORKING-STORAGE is the 01 area. This area contains a series of 01 group or elementary items. The 01 group items may be broken down in a manner similar to the record descriptions in the FILE SECTION. In the example above, a group item named WS-HEADING-LINE is defined then broken down into two FILLERS. The first FILLER simply takes up 64 characters of storage and initializes them to spaces. The second FILLER takes up 69 characters of storage and initializes the first 5 to the characters TITLE. The other 64 characters of the 69 are initialized to spaces. Use of the value clause will be described in detail later. All items written as 77 entries can also be written as 01's. 01's can be elementary or group items.

EXAMPLE:

```

77  WS-LINE-COUNT                PIC S9(4)
                                   USAGE IS COMP
                                   VALUE IS ZEROS
                                   SYNC.

77  WS-PAGE-COUNT                PIC S9(4)
                                   USAGE IS COMP
                                   VALUE IS ZEROS
                                   SYNC.

77  WS-PERSON-COUNT              PIC S9(4)
                                   USAGE IS COMP
                                   VALUE IS ZEROS
                                   SYNC.

```

This could be coded as:

```

01  WS-COUNTERS.
    05  WS-LINE-COUNT            PIC S9(4)
                                   USAGE IS COMP
                                   VALUE IS ZEROS
                                   SYNC.

    05  WS-PAGE-COUNT            PIC S9(4)
                                   USAGE IS COMP
                                   VALUE IS ZEROS
                                   SYNC.

    05  WS-PERSON-COUNT          PIC S9(4)
                                   USAGE IS COMP
                                   VALUE IS ZEROS
                                   SYNC.

```

176

or also as:

```

Ø WS-LINE-COUNT          PIC S9(4)
                           USAGE IS COMP
                           VALUE IS ZEROS
                           SYNC.

Ø1 WS-PAGE-COUNT         PIC S9(4)
                           USAGE IS COMP
                           VALUE IS ZEROS
                           SYNC.

Ø1 WS-PERSON-COUNT       PIC S9(4)
                           USAGE IS COMP
                           VALUE IS ZEROS
                           SYNC.

```

Army Standard: The clauses included with a data item description should be in the following order:

```

REDEFINES
OCCURS
PICTURE (PIC)
USAGE
VALUE
JUSTIFIED
SYNCHRONIZED (SYNC).

```

All clauses mentioned above will be described later in detail.

All three methods yield the same result as far as defining the data items and making them available for use in the PROCEDURE DIVISION. The difference lies in the resultant storage allocation. It is enough for the beginning programmer to know that for this type of data item the 77 is the most efficient and preferable of the three.

If the programmer needs a numeric field and if that field will only be used for mathematical computation (not part of an input or output record, i.e., linecounter), the programmer may specify that the USAGE of the numeric field is COMPUTATIONAL. This may be abbreviated as COMP. COMP fields are much more efficient for integer arithmetic than simple PIC 9 fields. You may not put in a V in the PICTURE. To print these fields is very time consuming, therefore, these should only be used for internal program counters. If the programmer chooses to specify COMPUTATIONAL (for efficiency purposes), he should guarantee that this field begin on a correct boundry (A COMPUTATIONAL field, on IBM SYSTEM/360, is a numeric field which should begin either on a half or full word boundry). To direct that this field is to start on a correct boundry, the programmer specifies SYNCHRONIZED (which may be abbreviated SYNC). See the above examples for this.

Army Standard: SYNCHRONIZED and COMPUTATIONAL are preferred to be abbreviated.

NOTE: A numeric field which is COMPUTATIONAL must include S (sign) as the

leftmost character in the picture clause.

If SYNCHRONIZED is used to describe a field which does not need to be aligned on a boundary (e.g., PICTURE X), it is ignored.

In the following examples, WS-AGE, not being a COMPUTATIONAL field, does not have to be aligned on any boundry; thus SYNCHRONIZED is ignored.

EXAMPLE:

```

Ø1 WS-CARD-K.D.
  Ø5 WS-NAME.
    1Ø WS-LAST-NAME          PIC X(1Ø).
    1Ø WS-FIRST-NAME         PIC X(1Ø).
  Ø5 WS-AGE                  PIC 999
                                SYNC.

```

If WS-AGE were not to be used as either a mathematical computation in the program or a numeric edited field on output, it could be defined as an alphanumeric (X) field:

```

Ø5 WS-AGE                    PIC X(3).

```

Note that each level entry ends with a period.

Any elementary item in WORKING-STORAGE (item with a PICTURE) may be assigned an initial value via the VALUE IS clause. Group items cannot be assigned a value and any data item under a REDEFINES clause cannot have a VALUE. REDEFINES is discussed in detail later. The only thing to keep in mind when assigning a VALUE to something is to remember how that thing is described in its PICTURE. Numeric items cannot be assigned an alphanumeric VALUE. Likewise, alphanumeric items should not be assigned a numeric literal VALUE; however, this is more acceptable than the first case. For example, one would not want to describe a counter as PIC 999 then try to initialize that counter to spaces. Spaces are not numbers and will not compute. COBOL has made things a little easier for programmers to initialize variables. There exist a number of reserved words that represent specific values when associated with a data item. ZEROS is used with numeric PICTURES. SPACES is used with alphanumeric PICTURES.

EXAMPLES (Not in the order that they would be in a program):

```

Ø1 WS-BLANK-LINE              PIC X(133)
                                VALUE IS SPACES.
Ø1 WS-COUNTER                 PIC 9(3)
                                VALUE IS ZEROS.
77 WS-LINE-COUNT              PIC S9(4)
                                VALUE IS ZEROS.
Ø1 WS-TITLE-1                 PIC X(5)
                                VALUE IS 'TITLE'.
77 WS-PAGE-COUNT              PIC 999
                                VALUE IS Ø.

```

The first example defines a 133 character block of memory and initializes it to spaces. The second example defines a 3 character numeric data item and initializes it to zeros. The third example defines a signed numeric variable of 4 characters and initializes it to zeros. The S in front of the 9 indicates that this variable may be negative or positive. The absence of an S indicates that this variable will always be positive and any value that is moved to that item is made positive. The fourth example defines a 5 character alphanumeric item and is initialized to a value of 'TITLE'. This example uses another form of initialization: the literal. Literals can be either alphanumeric (enclosed in quotes) for X PICTUREs or numeric for 9's PICTUREs. PICTUREs are described in detail in the next section. If a numeric PICTURE has an S and a numeric literal is used, then that numeric literal must have a + or - in front of it.

EXAMPLE:

```

77  LINE-COUNT                PIC S9(4)
                                USAGE IS COMP
                                VALUE IS +100
                                SYNC.

```

It is important to remember that a literal, whether numeric or alphanumeric, cannot exceed the length of the associated PICTURE. It was mentioned before that a numeric value should not be assigned to an alphanumeric PICTURE. ZEROS cannot be associated with an X PICTURE. However, the following is legal and desirable sometimes:

EXAMPLE:

```

77  NUMERIC-LITERAL          PIC X(10)
                                VALUE IS '0123456789'.

```

This example defines a 10 character alphanumeric data item and initializes it to the character string 0123456789. Anything placed within the quotes of the VALUE clause in an alphanumeric literal is legal and allowable. They are just another string of characters as far as the compiler is concerned. It should be remembered that one still cannot do arithmetic with the above example because the PICTURE is alphanumeric. If the literal used with an alphanumeric item is shorter than the PICTURE, the literal is placed in the left of the field and padded to the right with spaces.

RESTRICTION OF THE VALUE CLAUSE: IT CAN ONLY BE USED IN THE WORKING-STORAGE SECTION; IT CANNOT BE USED IN THE FILE SECTION.

Numeric literals may contain numeric digits (0 thru 9), a sign if the PICTURE clause has an S and a decimal point to indicate proper decimal point alignment.

EXAMPLES:

```

77 WS-PAGE-COUNTER          PIC 999
                             VALUE IS 1.
77 WS-TAX-RATE              PIC SV99
                             VALUE IS +.18.
77 WS-TOTAL-TAX             PIC S9(4)V99
                             VALUE IS +0.

```

Note that even when the literal has a decimal point in it, a period still ends the data item description giving two periods in the description.

If the literal used with a numeric PICTURE is shorter than the PICTURE, then the literal is placed in the right of the field and padded to the left with zeros. The only exception to this is when a V is included in a numeric PICTURE.

EXAMPLE:

```

77 WS-TAX-RATE              PIC SV99
                             VALUE IS +.23.
77 WS-NET-PAY               PIC S9(5)V99
                             VALUE IS +25.

```

In the first example, TAX-RATE is defined as a two character numeric variable with the implied decimal point to the left of the two digits. There is no actual decimal point in the field, but arithmetic is done as though it existed. Any number moved to this field will be adjusted so the implied decimal point is in the appropriate place. For example, if one moved the integer 33 to WS-TAX-RATE, WS-TAX-RATE would have zeros in it because 33 implied 33.00 and WS-TAX-RATE only has places for the numbers to the right of the decimal point. However, if one moved 33.25 to WS-TAX-RATE, WS-TAX-RATE would contain 25 with the implied decimal point to the left of the numbers. Decimal points are not numbers so they cannot be physically in the field that has the number being used in the computation. The V is used only for adjusting the significance of the digits. If it is possible for a move or computation to result in losing high order digits, a message will be printed in the error listing warning the programmer of the possibility. In the second example, WS-NET-PAY is defined as having seven characters of memory and is initialized to a value of 0002500 with the right two digits implied to be decimals. Notice that after the adjustment was made for the decimal point, the field was padded in both directions with zeros.

EDITED FIELDS

An EDIT field is designed to put symbols such as \$, . * + - in the correct place to make a numeric field more understandable. The following symbols may appear in a report field:

```

List 1: S V 9
List 2: Z , $ . + - * B CR DB

```


If one or more symbols from List 2 are used in a PICTURE clause, the field becomes a numeric EDIT field and cannot be used in any mathematical computation. If only the symbols in List 1 are used, the field remains a numeric field and may be used in any mathematical computation. The only way to get information into a EDIT field is to MOVE a numeric field TO an EDIT field (this instruction is given in the PROCEDURE DIVISION). Usually EDIT fields are used as part of an output description for a print file.

CHARACTER MEANING

- Z Zero suppress non-significant leading zeros.
- , Put a comma here if there are any significant digits to the left.
- \$ If one occurrence, put a dollar sign here; if multiple occurrences, put a dollar sign to the left of the first significant digit.
- . Decimal point.
- + Print a plus sign if the number moved into the field is positive; print a minus sign if the field is negative.
- Print a space if the number moved into the field is positive; print a minus sign if the field is negative.
- * Check protection: replace non-significant leading zeros with asterisks.
- V Implied decimal position. This does not take up a print position or a column on input.
- S Keep track of the sign for numeric fields (do not use S if any character from list 2 appears in the PICTURE clause).
- B Generate a blank wherever it is inserted.
- CR Prints two spaces if the value of the number moved into the field is positive. Prints as CR if the value is negative.
- DB Prints two spaces if the value of the number moved into the field is positive. Prints as DB if the value is negative.

EXAMPLES OF EDITED MOVES:

<u>PICTURE</u>	<u>VALUE OF DATA</u>	<u>EDITED RESULT</u>
	V's are shown for explanation only	
\$\$\$\$.99	V12	\$.12
\$\$,\$\$9.99	500V25	\$500.25
\$\$,\$\$9.99	1500V25	\$1,500.25
\$Z,ZZ9.99	1500V25	\$1,500.25
\$Z,ZZ9.99	-V25	\$ 0.25
\$Z,ZZ9.99-	-V25	\$ 0.25-
\$Z,ZZ9.99-	1V50	\$ 1.50
\$*,**9.99-	25V75	***25.75
\$*,**9.99+	25V75	***25.75+
\$*,**9.99+	-25V75	***25.75-
\$*,**9.99BCR	25V75	***25.75
\$*,**9.99BCR	-25V75	***25.75 CR
\$*,**9.99CR	-25V75	***25.75CR
\$*,**9.99BDB	25V75	***25.75
\$*,**9.99BDB	-25V75	***25.75 DB
\$*,**9.99DB	-25V75	***25.75DB
\$+,+++9.99	25V75	\$ +25.75
\$+,+++9.99	-25V75	\$ -25.75
\$-,---9.99	25V75	\$ 25.75
\$-,---9.99	-25V75	\$ -25.75

ARRAYS (TABLES)

An array is nothing more than a series of identical data items. Instead of having to define all of them individually, COBOL allows one to say that one thing OCCURS n times. All that the programmer has to do is define what one of the things looks like. The things are then referenced by use of an index or a subscript. The index or subscript simply tells the compiler which of the things you want to look at.

EXAMPLE:

```

01 WS-TABLE.
   05 WS-TABLE-ENTRY                OCCURS 25 TIMES
                                   INDEXED BY TABLE-INDEX.
       10 WS-NAME                    PIC X(20).
       10 WS-SOC-SEC-NUM              PIC X(9).

```

This example defines an array named WS-TABLE made up of 25 WS-TABLE-ENTRY's. Each WS-TABLE-ENTRY is broken down into a WS-NAME and WS-SOC-SEC-NUM. The entire array is 580 characters long. $((20+9) \times 25)$ This array is indexed by WS-TABLE-INDEX. How to reference arrays in the PROCEDURE DIVISION is covered in the section on the PROCEDURE DIVISION.

RESTRICTIONS ON ARRAYS: The OCCURS clause cannot be used with 77 or 01 items. Any item containing an OCCURS clause or subordinate to an item containing an OCCURS clause cannot have a VALUE clause.

REDEFINES CLAUSE

The same area of memory may be described in more than one way by using the REDEFINES clause. Many times it is necessary to use a particular data item in several ways (alphanumeric or numeric). This is done with the REDEFINES clause.

EXAMPLE:

```

01 WS-DISK-RCD.
   05 WS-PAY          PIC S9(5)V99
                       VALUE ZEROS.
   05 WS-PAY-1        REDEFINES WS-PAY
                       PIC X(7).

01 WS-CARD-RCD.
   05 WS-NAME-1.
       10 WS-LAST-NAME-1 PIC X(12).
       10 WS-FIRST-NAME-1 PIC X(8).
   05 WS-NAME-2        REDEFINES WS-NAME-1.
       10 WS-FIRST-NAME-2 PIC X(8).
       10 WS-LAST-NAME-2 PIC X(12).

```

In the first example, WS-PAY is simply REDEFINED as alphanumeric by WS-PAY-1. This type of redefinition is particularly useful when checking the validity of numeric fields. More of this will be covered in the discussion of the PROCEDURE DIVISION. In the second example, the storage assigned to WS-NAME-1 is defined in two ways: WS-LAST-NAME then WS-FIRST-NAME and WS-FIRST-NAME then WS-LAST-NAME. When the data coming into the program can be of several different forms, the REDEFINES clause is very helpful.

RESTRICTIONS ON THE REDEFINES CLAUSE:

1. Redefinition within the elements subordinate to an OCCURS clause will generate a warning.
2. Be sure to make the two items (the object and subject of the REDEFINES) the same length.
3. REDEFINES cannot be specified in an 01 entry in the FILE SECTION.
4. If A REDEFINES B, then both A and B must have the same level number and there must be no intervening level numbers having a level number equal to or less than A and B's level numbers. For example, if A and B were both 05

184

entries, then no entry with 05 or less could be between A and B. In example 2 above, there are 10's between NAME-1 and NAME-2. Both NAME-1 and NAME-2 are 05 entries.

88 LEVEL ITEMS

An 88 level item may be defined under any data item, both elementary and group level. It has no PICTURE clause. Instead it is used to describe possible values or ranges of values and to associate a name with the description.

EXAMPLE:

```
05 IT01-TRANSACTION-CODE    PIC X.
   88 ADDITION              VALUE 'A'.
   88 CHANGE                VALUE 'C'.
   88 DELETION              VALUE 'D'.
```

In this example, the elementary data item IT01-TRANSACTION-CODE is described as having the possible value of A, C, or D. If one of these values occurs, the name associated with them is true. See the discussion of the IF statement to see how to use the 88 in the PROCEDURE DIVISION. If any other value other than the ones described in the 88's occurs, none of the 88's are true, but no error condition is raised.

EXAMPLE:

```
01 WS-PAY                  PIC 9(6)V99.
   88 SMALL                VALUE .00 THRU 100.00.
   88 MEDIUM              VALUE 100.01 THRU 300.00.
   88 LARGE                VALUE 300.01 THRU 1000.00.
   88 VERY-LARGE          VALUE 1000.01 THRU 999999.99
```

Note in the first example that the values associated with the IT01-TRANSACTION-CODE are enclosed in quotes because the PICTURE of IT01-TRANSACTION-CODE is alphanumeric. In the second example, the VALUES are numeric literals because the PICTURE of WS-PAY is numeric. 88 level entries can be in both the FILE SECTION and the WORKING-STORAGE SECTION. They should not be used under 77 level items. Ranges may be used with alphanumeric items, but care should be exercised.

EXAMPLE:

```
05 WS-DIVISION-CODE        PIC XX.
   88 DIVISION-1          VALUE '01' THRU '02'.
   88 DIVISION-2          VALUE '03' THRU '04'.
```

What is being described is the internal machine representation of '01' thru the internal representation of '02' and all intervening characters. This type of 88 is very machine dependent and can be the cause of obscure errors in processing.

THE PROCEDURE DIVISION

The PROCEDURE DIVISION contains the instructions that manipulate the data described in the DATA DIVISION. The PROCEDURE DIVISION is made up of a series of paragraphs. Each paragraph begins with a name (the paragraph name) followed by a period followed by the body of the paragraph. The body is made up of one or more COBOL PROCEDURE DIVISION sentences. A COBOL sentence is one or more COBOL verbs and their associated parts. A sentence is ended by a period. All paragraph names begin in margin A. All sentences begin in margin B.

Army Standard: All paragraph names will be preceded by a four digit number. Paragraphs will be in ascending order by this number. The numbers will be incremented by 10 to allow insertion of new paragraphs in the future.

EXAMPLE:

PROCEDURE DIVISION.

0010-PARA-NAME.

OPEN INPUT CARD-READER

OUTPUT PRINTER.

PERFORM 0020-READ-AND-PRINT THRU 0020-RAP-EXIT

UNTIL WS-EOF-SWITCH = 'OFF'.

CLOSE CARD-READER,

PRINTER.

STOP RUN.

0020-READ-AND-PRINT.

READ CARD-READER

AT END MOVE 'OFF' TO WS-EOF-SWITCH

GO TO 0020-RAP-EXIT.

MOVE CARD-RECORD TO PRINT-RECORD.

WRITE PRINT-RECORD AFTER ADVANCING 2 LINES.

0020-RAP-EXIT. EXIT.

In the above example, there are three paragraphs. The first starts with 0010-PARA-NAME and ends with STOP RUN. The second paragraph begins with 0020-READ-AND-PRINT and ends with WRITE PRINT-RECORD AFTER ADVANCING 2 LINES. The third begins with 0020-RAP-EXIT and ends with EXIT (this has only one sentence in it). EXIT will be explained later, but for now, the EXIT paragraph is considered to be the end of the preceding paragraph. What this program actually does is immaterial at this point. Each of the verbs (OPEN, PERFORM, CLOSE, STOP, READ, MOVE, GO, WRITE) will be discussed in detail later. When the PROCEDURE DIVISION is executed, the first statement after PROCEDURE DIVISION is executed first. Because of this, no paragraph name is needed immediately after PROCEDURE DIVISION. ANSI requires para name. Normally paragraphs represent blocks of code that do one job. In the example, everything from 0020-READ-AND-PRINT thru 0020-RAP-EXIT is involved in the reading and writing of data records. Everything from the PROCEDURE DIVISION thru the STOP RUN is involved in preparing the files to be read and written, controlling the reading and writing, then cleaning up after it has all been done. Each paragraph has its own job and doesn't depend on anything else to do that job for it.

The PROCEDURE DIVISION's verbs have been divided into three classes:

1. Process verbs - words which initiate manipulation of data in the program.
2. I/O verbs - words which initiate data transfer from the outside world into the program or vice versa.
3. Logic control verbs - words which affect the order of statement execution.

Each of these categories of verbs are discussed below.

PROCESS VERBS

MOVE

The form of the MOVE verb is:

MOVE {data-name-1} TO data-name-2 [data-name-3 ...]
 {literal}

This verb copies information from one data-name or literal to one or more data-names. There are several restrictions on moving data from one area to another that are connected to the PICTUREs of the respective data-items. See Table 1. An example of the MOVE statement is:

EXAMPLE:

MOVE IMØ1-NAME TO WS-NAME.

MOVE IMØ1-DIVISION TO WS-DIVISION
 WS-DIVISION HOLD.

In the first example, the data-item IMØ1-NAME is copied to WS-NAME. When the move is complete, both IMØ1-NAME and WS-NAME contain the information that IMØ1-NAME had contained before the MOVE was executed. In the second example, the information in IMØ1-DIVISION is copied to both WS-DIVISION and WS-DIVISION HOLD. When the MOVE is complete, all three data items have the same information in them.

Another form of the MOVE statement is:

MOVE {CORRESPONDING} group-item-1. TO group-item-2.
 {CORR}

This form of the MOVE verb copies all of the data-names under group-item-1 that have the same data-names under group-item-2. When the MOVE CORRESPONDING is complete, anything from all of the data-items under group-item-1 to none of the data-items under group-item-1 may have been moved.

Army Standard: Since non-unique names are not allowed and the move corresponding depends on non-unique names, the move corresponding statement is not allowed.

EXAMPLE:

FILE SECTION.

```
.
Ø1 IMØ1-MASTER-RCD.
  Ø5 IMØ1-NAME          PIC X(2Ø).
  Ø5 IMØ1-SSN          PIC X(9).
  Ø5 IMØ1-RANK          PIC X(3).
  Ø5 FILLER             PIC X(48).
```

WORKING-STORAGE SECTION.

```
.
Ø1 WS-DETAIL-LINE-1.
  Ø5 FILLER              PIC X(48)
                          VALUE IS SPACES.
  Ø5 WS-NAME             PIC X(2Ø).
  Ø5 FILLER              PIC X(3)
                          VALUE IS SPACES.
  Ø5 WS-RANK             PIC X(3).
  Ø5 FILLER              PIC X(3)
                          VALUE IS SPACES.
  Ø5 WS-SSN              PIC X(9).
  Ø5 FILLER              PIC X(47)
                          VALUE IS SPACES.

Ø1 WS-DETAIL-LINE-2.
  Ø5 FILLER              PIC X(48)
                          VALUE IS SPACES.
  Ø5 IMØ1-NAME           PIC X(23).
  Ø5 IMØ1-RANK           PIC X(6).
  Ø5 IMØ1-SSN           PIC X(56).
```

PROCEDURE DIVISION.

```
.
MOVE CORRESPONDING IMØ1-MASTER-RCD TO WS-DETAIL-LINE-1.
MOVE CORRESPONDING IMØ1-MASTER-RCD TO WS-DETAIL-LINE-2.
```

In the first MOVE CORRESPONDING statement, nothing would have been moved and no error message generated. This is one weakness of the MOVE CORRESPONDING statement. None of the data-items are named identically under group-item-1 and group-item-2. In the second MOVE CORRESPONDING, all of the data-item in MASTER-RECORD are moved in DETAIL-LINE-2. Note that instead of defining all of the FILLERS between the data-items as in DETAIL-LINE-1, the data-items in DETAIL-LINE-2 are simply defined as containing those FILLERS. When the

IMØ1-NAME defined in IMØ1-MASTER-RECORD is moved to the IMØ1-NAME defined in WS-DETAIL-LINE-2, the information is moved into the left most of the 23 character receiving field and the remaining 3 characters are filled with blanks. Similar actions accompany the other two moves. This type of definition is less efficient than the first because the trailing characters in all of the receiving fields are filled with blanks each time the move statement is executed. In the first example, the blanks are put in only once during the compilation and remain there all during the execution of the program with no modification. It is normally a good idea to make receiving fields the same size as sending fields.

NOTE: If any of the data item names are defined with the same data name in two or more areas and need to be referenced individually, they may be referenced by qualifying the item with the qualifier OF.

EXAMPLE:

MOVE RANK OF MASTER-RECORD TO RANK-HOLD.

Army Standard: Qualification of data names will not be used.

TABLE 1

LEGAL MOVES:

1. (A) an alphabetic field may be moved to another alphabetic field. The movement is from the leftmost byte of the receiving field to the right. If the receiving field is larger, the remaining positions will be filled with spaces. If the receiving field is shorter than the sending field, truncation of the right most position(s) will occur.
2. (X) an alphanumeric field may be moved to another alphanumeric field. The same rules apply here as they do in an alphabetic field.
3. (9) a numeric field may be moved to another numeric field. However, the movement is accomplished by decimal point alignment. If the receiving field is smaller, truncation will occur and if the receiving field is larger, zeros will fill in the remaining core area. Exactly how the truncation takes place will be explained later.
4. A numeric or alphabetic field may be moved to an alphanumeric field. When numeric (PIC 9's) data fields are moved to alphanumeric (PIC X's) data fields, the move is treated as an alphanumeric move and follows the rules shown under rule 1.
5. An edited field may be moved to an alphanumeric field.
6. A numeric field may be moved to an edited field.
7. An alphabetic field may be moved to an alphanumeric edited field.

EXAMPLE:

MOVE A-FIELD TO B-FIELD.

If B-Field were alphanumeric, A-Field could be alphabetic, alphanumeric or edited.

If B-FIELD were numeric, A-FIELD could only be numeric.

If B-FIELD were alphabetic, A-FIELD could only be alphabetic.

If B-FIELD were edited, A-FIELD could only be numeric.

In the above examples, A-FIELD remains unchanged; B-FIELD becomes a "copy" of A-FIELD.

EXAMPLE:

MOVE SPACES TO ORØ1-PRINTER-RCD.

In this case, ORØ1-PRINTER-RCD must be defined as PICTURE X (alphanumeric) or PICTURE A (alphabetic), because spaces is recognized by the compiler as 1 or more "blanks", which is a character, not a number, and can be placed only into a PICTURE (A) or (X) fields.

NOTE: All group level items (items which are broken down into smaller parts) are handled as PICTURE X fields regardless of the PICTURE of the subordinate elements of the field are described.

EXAMPLE:

```
Ø1 WS-OUT-RCD.
   Ø5 WS-NUMBER      PIC 9(6).
   Ø5 FILLER          PIC X(127).
```

In this example, WS-OUT-RCD is alphanumeric (PICTURE (X)) because it is a group item. Its subordinate parts are WS-NUMBER and FILLER, both elementary items because they have PICTURE clauses. WS-OUT-RCD's length is the sum of its parts (133 characters). The programmer can legally write MOVE SPACES TO WS-OUT-RCD because WS-OUT-RCD is a group item and therefore alphanumeric even though the first six positions are defined as numeric. If this move was done, then WS-NUMBER cannot be used in arithmetic computations because the information in it is not numeric and will cause a 'data exception' (trying to use non-numeric data as though it were numeric). Numeric information must be moved to WS-NUMBER before WS-NUMBER can be used as a number.

If the two items are alphanumeric and not the same size, one of two things will happen:

1. If the smaller item is moved to the larger, the data in the smaller item is put as far left as possible in the larger field. The remaining characters of the larger item will be filled with spaces.

2. If the larger item is moved to the smaller, the smaller item is filled up with the information from the larger starting at the left of both fields. The extra characters in the larger field are ignored (truncated).

EXAMPLE:

```

01  IM01-CARD.
    05  IM01-NAME          PIC X(20).
    05  IM01-SSN           PIC X(9).
    05  IM01-ADDRESS       PIC X(40).
    05  FILLER             PIC X(11).

```

```

01  WS-DETAIL-LINE.
    05  FILLER             PIC X(33).
    05  WS-NAME            PIC X(25).
    05  WS-SSN             PIC X(15).
    05  WS-ADDRESS        PIC X(30).
    05  FILLER             PIC X(32).

```

```

MOVE IM01-NAME TO WS-NAME.
MOVE IM01-SSN TO WS-SSN.
MOVE IM01-ADDRESS TO WS-ADDRESS.

```

IM01-NAME is moved to the left most twenty characters of WS-NAME and the remaining five characters are filled with spaces. IM01-SSN is likewise put in the leftmost nine characters of WS-SSN and the remaining six characters are filled with spaces. Since IM01-ADDRESS is larger than WS-ADDRESS, only the leftmost thirty characters of IM01-ADDRESS are moved to WS-ADDRESS.

All numeric fields have a decimal point either implied (with a 'V'), assumed (if no other decimal point is given, it defaults to the right of the rightmost digit), or indented (as a '.'). Movement from one numeric field to another takes place after decimal point alignment. One or both of these cases may apply.

1. After decimal point alignment, the digits are moved from the sending field into the receiving field, beginning at the left of the decimal point and proceeding left. If the sending field has more digits to the left of the decimal point than the receiving field, only as many as will fit are moved. The rest are truncated. If the sending field has fewer digits to the left of the decimal point than the receiving field, then the extra digits are filled with zeros.

2. If the sending field has more digits to the right of the decimal point than the receiving field, then only as many as will fit are moved. The rest are truncated. If the sending field has fewer digits to the right of the decimal point than the receiving field, then the extra digits are filled with zeros.

EXAMPLES:

```

77 WS-NUM-1          PIC 99V9
                        VALUE 25.1.
77 WS-NUM-2          PIC 999V99.
77 WS-NUM-3          PIC V999.
77 WS-NUM-4          PIC 99.

```

```

MOVE WS-NUM-1 to WS-NUM-2,
                  WS-NUM-3,
                  WS-NUM-4.

```

After the moves, the value of WS-NUM-1 is 25.1, WS-NUM-2 is 025.10, WS-NUM-3 is .100, and WS-NUM-4 is 25.

To the left of the decimal point, the data is right justified. Any excess high order positions are filled with 0's. To the right of the decimal point, they are left justified. Any excess low order positions are filled with 0's. Truncation occurs as follows: To the left of the decimal point, the high order is truncated. To the right of decimal point, the low order is truncated.

ADD

There are three forms of the ADD verb. The first form is normally used to add one or more things to a sum.

```

ADD {data-name-1      } [data-name-2      ...
    {numeric-literal-1} [numeric-literal-2
    {data-name-m      } TO data-name-n [ROUNDED]
    {numeric-literal-m}

```

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

EXAMPLE 1:

```
ADD WS-FICA-TAX WS-CITY-TAX WS-LOCAL-TAX WS-STATE-TAX TO WS-TOTAL-TAX.
```

EXAMPLE 2:

```
ADD 1 to WS-LINE-COUNT.
```

The first example adds four individual taxes to the total already in WS-TOTAL-TAX yielding a new total. This list of taxes could also be added to several other things in the same statement by simply listing the things after WS-TOTAL-TAX. Each of the resultant fields may have the ROUNDED clause included after it. If ROUNDED is used, the result is rounded according to the receiving field PICTURE. After decimal point alignment, the number of places to the right of the decimal point of the result of an arithmetic computation is compared with the number of places allowed for the result in the receiving field. If more

places are in the result than can be accommodated for in the receiving field, normally the extra places are simply discarded (truncated). If ROUNDED is specified, then the most significant digit of the excess is examined. If this digit is 5 or more, then the least significant digit of the part that is going to be used will be incremented by one. If this digit is 4 or less, then the number remains the same and the unneeded low order digits are discarded. If ON SIZE ERROR is specified after the list of receiving fields, then if the result of the computation is too large to fit in the specified PICTURE, the list of verbs after the ON SIZE ERROR is executed. Whether the ON SIZE ERROR clause is specified or not and a high order truncation condition occurs, the truncated result is still put in the receiving fields. In the second example, WS-LINE-COUNT is incremented by 1.

Army Standard: Use of ON SIZE ERROR and ROUNDED should be avoided when possible.

The second form of the ADD verb is used to add two or more things into one or more receiving fields. ROUNDED and ON SIZE ERROR may be specified for each of the receiving fields causing the same treatment as in the first form of the ADD verb.

ADD {data-name-1 data-name-2 [...] } GIVING data-name-m [ROUNDED]
 {numeric-literal-1 numeric-literal-2

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

EXAMPLE:

```
ADD IM01-NUMBER-OF-DEPENDENTS 1 GIVING WS-FAMILY-SIZE,
    ON SIZE ERROR DISPLAY 'FAMILY SIZE TOO BIG'
    GO TO 0100-PARA-EXIT.
```

In this example, IM01-NUMBER-OF-DEPENDENTS and the numeric-literal 1 are added together and the result is placed into WS-FAMILY-SIZE. If WS-FAMILY-SIZE cannot contain the number of digits of the result of the computation, then an ON SIZE ERROR condition exists and the verbs after ON SIZE ERROR until the next period are executed. If ON SIZE ERROR had not been specified and the condition arose during the computation, the result would be truncated and the next statement executed. An advantage of the GIVING option is that none of the original values, except the result, are changed by the ADD.

The third form of the ADD verb adds all of the items under group-item-1 to all of the identically named items under group-item-2. ROUNDED and ON SIZE ERROR operate identically as stated before.

ADD CORRESPONDING group-item-1 TO group-item-2 [ROUNDED]
 CORR

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

EXAMPLE:

ADD CORRESPONDING IMØ1-MASTER-PAY-RECORD TO WS-MASTER-PAY-TOTALS.

This example will add all of the items under IMØ1-MASTER-PAY-RECORD to the identically named items under WS-MASTER-PAY-TOTALS truncating the results if necessary.

Army Standard: Add corresponding will not be used.

SUBTRACT

SUBTRACT has three forms. They are:

SUBTRACT { data-name-1
numeric-literal-1 } [data-name-2
numeric-literal-2 ...] FROM data-name-m [ROUNDED

[data-name-n [ROUNDED] ...]

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

SUBTRACT { data-name-1 data-name-2 ... } FROM data-name-m
numeric-literal-1 numeric-literal-2 numeric-literal-m

GIVING data-name-n [ROUNDED]

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

SUBTRACT { CORRESPONDING } group-item-1 FROM group-item-2 [ROUNDED]
(CORR)

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

In the first form of the SUBTRACT verb, one or more items are added up then the total is subtracted from the item or items following the FROM. ROUNDED and ON SIZE ERROR work identically to the ADD statement.

Army Standard: Use of ON SIZE ERROR and ROUNDED should be avoided where possible.

EXAMPLES:

SUBTRACT 1 FROM WS-TOTAL-PROPLE.

SUBTRACT WS-FICA-TAX WS-SUC-SEC-TAX FROM WS-GROSS-PAY ROUNDED

ON SIZE ERROR MOVE '1' TO WS-ERROR-FLAG

GO TO Ø15Ø-PARA-EXIT.

194

In the second form of the SUBTRACT verb, one or more items are added up then the total is subtracted from the item after the FROM clause and the final result is put into the one or more data-items following the GIVING clause. ROUNDED and ON SIZE ERROR work identically to the ADD statement.

EXAMPLES:

SUBTRACT 1 FROM WS-FAMILY-SIZE GIVING WS-NUMBER-OF-DEPENDENTS.
SUBTRACT WS-FICA-TAX WS-SOC-SEC-TAX FROM WS-GROSS-PAY GIVING WS-NET-PAY.

In both the examples, the item following the FROM is not changed by the computation.

In the third form of the SUBTRACT verb, all of the items under group-item-1 are subtracted from the identically named items under group-item-2. ROUNDED and ON SIZE ERROR work identically to the ADD statement.

EXAMPLE:

SUBTRACT CORR IM03-INVENTORY-ADJUSTMENT-RECORD FROM
WS-TOTAL-INVENTORY-DISCREPANCY.

Army Standard: Subtract corresponding will not be used.

MULTIPLY

This verb is used to multiply one by another. It has two forms.

MULTIPLY {data-name-1
numeric-literal-1} BY data-name-2 [ROUNDED]

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
ELSE (period)]

MULTIPLY {data-name-1
numeric-literal-1} BY {data-name-2
numeric-literal-2} GIVING data-name-3 [ROUNDED]

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
ELSE (period)]

In the first form of the MULTIPLY verb, a number (either a numeric-literal or a data-name) is multiplied by another data-name and the result is put in the data-name after BY. ROUNDED and ON SIZE ERROR work in the same manner as in the ADD statement.

Army Standard: Use of ROUNDED and ON SIZE ERROR should be avoided where possible.

EXAMPLE:

MULTIPLY WS-TAX-RATE BY WS-GROSS-PAY.

This example multiplies the contents of WS-TAX-RATE by the contents of WS-GROSS-PAY and puts the result in WS-GROSS-PAY.

The second form of the MULTIPLY verb is used to multiply two values together putting the result into a data-name. ROUNDED and ON SIZE ERROR work in the same way as in the ADD statement.

EXAMPLE:

```
MULTIPLY WS-GROSS-PAY BY WS-TAX-RATE GIVING WS-FEDERAL-TAX ROUNDED
ON SIZE ERROR DISPLAY 'FEDERAL TAX OVERFLOW'
GO TO 0210-PARA-EXIT.
```

DIVIDE

The DIVIDE statement has two forms:

DIVIDE {data-name-1
numeric-literal-1} INTO data-name-2 [ROUNDED]

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
ELSE (period)]

DIVIDE {data-name-1
numeric-literal-1} {INTO
BY} {data-name-2
numeric-literal-2} GIVING data-name-3 [ROUNDED]

[REMAINDER data-name-4]

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-
ELSE (period)]

The first form of the DIVIDE verb is used to divide a number or data-name into another data-name. ROUNDED and ON SIZE ERROR work the same way as in the ADD statement. The result is left in the second data-name.

Army Standard: Use of ROUNDED and ON SIZE ERROR should be avoided where possible.

EXAMPLES:

```
DIVIDE 10 INTO WS-PAGE-TOTAL.
DIVIDE WS-TAX INTO WS-PAY ROUNDED.
```

In both examples, the result is left in the second variable (after the INTO).

The second form of the DIVIDE verb divides a number or data-name into or by another number or data-name giving a third number or data-name. ROUNDED and ON SIZE ERROR work in the same manner as in the ADD statement.

196

EXAMPLES:

DIVIDE WS-TAX INTO WS-PAY GIVING WS-TAX-RATE.
DIVIDE A BY B GIVING C ROUNDED.

In both examples, the result is placed in the variable after the GIVING leaving the first two variables unchanged.

COMPUTE

COMPUTE identifier-1 [ROUNDED] = {arithmetic-expression
 identifier-2
 literal-1}

[ON SIZE ERROR any series of COBOL verbs and associated parts except IF-ELSE (period)]

The COMPUTE statement gives COBOL powers similar to that of FORTRAN: a complete arithmetic expression may be evaluated and the result placed in identifier-1.

There are five (5) arithmetic operators that may be used in arithmetic expressions. Each is represented by a specific character or character combination that must be preceded by a space and followed by a space, except that a sign (- or +) must not be preceded by a space when it follows a left parenthesis.

ARITHMETIC OPERATOR	MEANING
+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

Parenthesis may be used in arithmetic expressions to specify the order in which elements are to be evaluated. Expressions within parenthesis are evaluated first. When parenthesis are not used or parenthetical expressions are at the same level of inclusiveness, the following heirarchical order is implied:

1. - or + (indicating negative or positive number)
2. **
3. * and /
4. + and -

When the order of consecutive operations on the same heirarchical level is not completely specified by parentheses, the order of operation is from left to right.

ON SIZE ERROR and ROUNDED operate identically to the ADD statement.

Army Standard: Use of the ON SIZE ERROR and ROUNDED should be avoided where possible.

EXAMPLE: COMPUTE WS-RESULTS ROUNDED = (A + B - C * D) - (-E / F + (G * H ** I))

ORDER OF OPERATION:

H is raised to the Ith power
 G is multiplied by the result of (H**I)
 E is negated
 F is divided into E
 The result of -E/F is added to G*H**I
 C is multiplied by D
 A is added to B
 C*D is subtracted from A+B
 The result of -E/F+(G*H**I) is subtracted from the result of A+B-C*D
 This result is rounded up and the result is placed into WS-RESULTS.
 A B C D E F G H and I remain unchanged.

NOTE: Army naming convention is dropped for this example for illustrative clarity.

INPUT-OUTPUT

READ

The form of the READ statement is:

READ file-name

AT END any series of COBOL verbs and associated parts except the IF or IF-ELSE ended by a period.

This verb reads any sequential file (cards, disk, or tape file) and places the information that was read into the record description under the file-name in the FD. Each time this verb is executed, one record is read from the file. After the last record has been READ, the next time the READ statement is executed, the AT END series of verbs is executed. Care should be exercised to insure that once the AT END condition arises the file is never READ again except when another file immediately follows the one just processed. If a record was actually READ, the period (after the AT END series of verbs) is executed.

EXAMPLE:

```
READ CARD-FILE
  AT END MOVE 'OFF' TO WS-READ-END-OF-FILE-SWITCH
  GO TO 1450-PARA-EXIT.
MOVE IM03-NAME TO WS-NAME.
```

In the example, a card will be READ each time the READ statement is executed. After the last card is READ and the statement is executed once more, 'OFF' will be MOVED to WS-READ-END-OF-FILE-SWITCH and control will go to the paragraph exit. If the AT END condition did not occur when the READ was executed, the MOVE statement following the period after the GO TO 1450-PARA-EXIT would be executed. If the period after the GO TO 1450-PARA-EXIT was accidentally omitted, the MOVE statement would be considered part of the AT END process and the statement after the MOVE would not be executed following a good READ. Likewise, if the period was placed after the MOVE 'OFF' statement, then the GO TO 1450-PARA-EXIT would be executed for each good READ. Clearly, placement of period after AT END is important to program execution.

- Army Standard:
- (1) Only one READ per file is normally required. This should be in a one statement paragraph that is performed from the needed places in the program.
 - (2) Never execute a READ for a file which has encountered end-of-file unless it has been closed and reopened as input.

WRITE

A form of the WRITE verb is:

<u>WRITE</u> file-record	[<u>FROM</u> record-name]	<u>BEFORE</u>	ADVANCING	data-name	LINES.
		<u>AFTER</u>		integer	
				mnemonic-name	-

This form writes to the printer or card-punch. Each time the WRITE statement is executed, the information in the record under the file description is written to the device specified (either a card punch or line printer). If the FROM option is included, the information in the record-name is copied to the file-record under the file description and then written. If the punch is being written to, then the BEFORE option should be chosen if the programmer wants to determine which punch pocket the cards should be placed in after it is physically punched. A mnemonic name associated with the punch pocket in the SPECIAL-NAMES paragraph is used to select the actual pocket. If a printer is being written to, either BEFORE or AFTER may be used and the corresponding writing will take place. If BEFORE is specified, then the line being written will be written before the spacing. If AFTER is specified, then the line will be written after the spacing. Spacing is controlled by the ADVANCING clause. If a data-name is used, then it must be a non-negative elementary numeric item (normally a 77 entry) whose value lies between 0 and 99. If an integer is chosen, the same constraints on the values apply as with the data-name. If a mnemonic-name is chosen, then the mnemonic-name used must be defined in the SPECIAL-NAMES paragraph and associated with one of the printer carriage control channels.

EXAMPLES:

1. Write to the card punch:

SPECIAL-NAMES.

SØ1 IS TO-POCKET-1.

WRITE OPØ1-CARD-PUNCH BEFORE ADVANCING TO-POCKET-1.

or

WRITE OPØ1-CARD-PUNCH.

2. Write to the printer.

SPECIAL-NAMES.

CØ1 IS CHAN1.

WRITE ORØ1-PRINTER-RCD AFTER ADVANCING CHAN1.

or

WRITE ORØ1-PRINTER-RCD FROM WS-DETAIL-LINE AFTER ADVANCING 2 LINES.

or

77 WS-CARR-CONTROL PIC 99
VALUE 4.

WRITE ORØ1-PRINTER-RCD BEFORE ADVANCING WS-CARR-CONTROL LINES.

In the first example, TO-POCKET-1 is associated with pocket one (SØ1) in the card punch. The write statement forces the punched card to be placed into pocket 1 through use of the BEFORE ADVANCING TO-POCKET-1 clause. In the second case of writing to the card punch, the programmer simply allows the system to choose the pocket for the card to be placed in. Pocket selection is normally done when two or more types of cards are being punched and are to be kept separate. When the ADVANCING option is used, an extra character at the left of the record must be reserved for the system to use; e.g., to punch an 8Ø character record, 81 characters must be defined with the first character being a FILLER. When the BEFORE ADVANCING option is not used, then only 8Ø characters must be defined.

In the second example, CHAN1 is associated with printer carriage channel 1. Printer carriage channel 1 is almost always lined up with line six of the printer page. When, as in the first WRITE statement, the line is written AFTER ADVANCING CHAN1, the paper in the printer is advanced until printer carriage channel 1 is sensed then the line is written. This allows the

programer to begin headings at the top of a new page. In the second case, the information in WS-DETAIL-LINE is moved to ORØ1-PRINTER-RCD then ORØ1-PRINTER-RCD is written after double spacing past the last line. In the third case, a data item is used to determine the spacing. The value that the printer is advanced BEFORE or AFTER writing the line depending on the option chosen. As in the card punch, whenever the ADVANCING option is used, an extra character on the left of the line must be reserved for system use. The IBM 1403 printer has 132 printable positions. When ADVANCING is used on this device, 133 characters must be written. The first character of the 133 is used by the system and the remaining 132 are printed.

Another form of the WRITE statement is:

WRITE file-record [FROM record-name].

This form is used to write to tape files. There is no BEFORE or AFTER option. The FROM clause is optional.

EXAMPLES:

WRITE OTØ1-TAPE-FILE-RCD.

WRITE OTØ1-TAPE-FILE-RCD FROM WS-ADD-RECORD.

In the first example, all of the information to be written to the tape file had been moved into OTØ1-TAPE-FILE-RCD (the Ø1 record description under the FD for the tape file) prior to the WRITE. In the second case, the information had been moved from WS-ADD-RECORD to OTØ1-TAPE-FILE RCD prior to WRITE. WS-ADD-RECORD would normally be defined in the WORKING-STORAGE SECTION somewhere.

The last form of the WRITE statement is:

WRITE file-record [FROM record-name]

INVALID KEY a series of COBOL verbs and associated parts except IF-ELSE (period).

This form is used to write to disk files. File-record is the Ø1 under the FD for the disk file. The FROM record-name is optional and is used identically as the other two examples. INVALID KEY is used to tell the programer when the disk file is full and no more records can be written to it. Disk files have a definite amount of space to hold the information they are given. When that space is all used, then the program has to be informed of it. This is done via the INVALID KEY clause. When the file is full and one more record is written, the series of COBOL verbs and associated parts after the INVALID KEY is executed. An IF verb cannot be one of the verbs used in the series.

Army Standard: (1) Only one WRITE for each record of a different size should be in a program. Each WRITE should be a one statement paragraph that is PERFORMed from the appropriate places.

- (2) For files with several types of records, it is best to define them in WORKING-STORAGE and move them to the record area just before the WRITE.

DISPLAY

DISPLAY is a useful verb for debugging the program. It should be used to print intermediate results and values during the run of a program. It relieves the programmer of defining a printer file during the debugging process and then having to pull all of the cards defining that file out of the program after the debugging phase is over. Its form is:

DISPLAY a series of literals (things enclosed in quotes) and data-names.

If the series is over 120 characters in length, only 120 characters will be printed on a line and the rest will be printed on the next line.

EXAMPLE:

DISPLAY 'NEW MASTER RECORD IS: ' OM01-NEW-MASTER-RECORD.

This example will print the literal 'NEW MASTER RECORD IS: ' followed by the characters in the OM01-NEW-MASTER-RECORD all on one line. Because the literal is 22 characters long, the number of characters in OM01-NEW-MASTER-RECORD that can be printed on the printer is 98. If OM01-NEW-MASTER-RECORD is longer than 98 characters, then only the first 98 characters are printed on this line and the rest are printed on the next line. Each time a DISPLAY is executed, the contents of the series of literals and variables are begun on a new line single spaced after the last line. There is no way to go to the top of a page with a DISPLAY statement. Double spacing can be achieved by simply displaying a blank before or after each real DISPLAY.

EXAMPLE:

DISPLAY ' '.
DISPLAY 'TOTAL NUMBER OF PEOPLE IS: ' WS-PEOPLE-TOTAL.

This example prints a blank line then a line giving the total people message.

Army Standard: Display should only be used as a debugging tool except in communications with the operator.

EXHIBIT (IBM Extension)

The EXHIBIT verb is another debugging tool. Its form is:

EXHIBIT NAMED series of data-names and literals.
CHANGED



This verb formats each of the data-names into a DATA-NAME=value, form. These are spread out across the page. If the list is too long to fit on one line, then it is carried over to the next line. When the EXHIBIT NAMED form is used, then every data-name in the list is listed each time the EXHIBIT is executed. When the EXHIBIT CHANGED form is used, all of them are reported the first time the verb is executed. The next time the verb is executed, only those data-names that have changed since the last execution are listed.

Army Standard: EXHIBIT should only be used as a debugging tool.

EXAMPLE:

EXHIBIT NAMED WS-DIVISION-HOLD WS-LINE-COUNT.

EXHIBIT CHANGED WS-DIVISION-HOLD WS-LINE-COUNT.

In the first example, WS-DIVISION-HOLD and WS-LINE-COUNT will be listed each time the statement is executed. In the second example, both WS-DIVISION-HOLD and WS-LINE-COUNT are printed the first time the EXHIBIT is executed. From then on whenever the statement is executed, only the data item that has changed (one or both) will be printed. If neither one has changed, then nothing is printed.

OPEN

The OPEN verb is used to prepare the files for input or output. Execution of this verb must precede the execution of any READ or WRITE statement. Its form is:

OPEN INPUT one or more file-names to be opened as either INPUT or OUTPUT
 OUTPUT

INPUT one or more file-names to be opened as either INPUT or OUTPUT...]
 OUTPUT

EXAMPLES:

OPEN INPUT PAY-MASTER-FILE.
OPEN INPUT CARD-READER DISK-FILE
 OUTPUT PRINTER NEW-DISK-FILE.
OPEN OUTPUT CARD-PUNCH PRINT-FILE.

The first example opens the PAY-MASTER-FILE as an input file and prepares that file for processing. The second example opens the CARD-READER and DISK-FILE as input files then opens the PRINTER and NEW-DISK-FILE as output files. The third example opens the CARD-PUNCH and PRINT-FILE as output files. Care should be exercised not to open a file more than once unless a CLOSE (described next) for that file has been executed in the interim. It is completely legal and necessary sometimes to OPEN a file as output then when processing is complete on that file, to close it and open it as input. All of the file names must be described in an FD in the FILE SECTION.

Army Standard: As many files as possible should be opened by one OPEN statement.

CLOSE

The CLOSE verb is used to disassociate a file with a program. Before the program is finished execution, all of the files currently open should be closed. If it is necessary to access a file as both input and output at different times, a CLOSE statement must be executed between the two types of access. The form of the CLOSE verb is:

CLOSE one or more file-names

EXAMPLES:

CLOSE INPUT-MASTER PRINTER.
CLOSE PRINTER.
CLOSE CARD-READER
DISK-FILE.

All of the examples simply CLOSE the list of files after the CLOSE verb.

Army Standard: As many files as possible should be closed by one CLOSE statement.

LOGIC FLOW CONTROL VERBS

Paragraph names, while not verbs, are used by the logic control verbs in branching. They are A margin entries and are used to identify logically significant points in the program. They follow the same naming convention as data names; however, they may be completely numeric also. This is not recommended. They are ended with a period.

EXAMPLES:

Ø17Ø-READ-AND-PRINT.

.

Ø17Ø-RAP-EXIT.

IF is used to make decisions during the execution of the program. Its form is:

IF condition any series of legal COBOL verbs and associated parts.

204
EXAMPLE:

```
IF IM01-SOC-SEC-NUM EQUAL TO IT01-SOC-SEC-NUM
  PERFORM 0100-UPDATE-ROUTINE THRU 0100-UPDATE-EXIT
  PERFORM 0200-READ-NEW-MASTER THRU 0200-READ-EXIT
  PERFORM 0300-READ-UPDATE THRU 0300-READ-EXIT.
```

In the above example, if IM01-SOC-SEC-NUM is equal to IT01-SOC-SEC-NUM, then the 0100-UPDATE-ROUTINE is PERFORMed, the 0200-READ-NEW-MASTER routine is PERFORMed, then the 0300-READ-UPDATE routine is PERFORMed. The entire series is executed only if the condition is true. If the condition is not true, then the sentence after the period ending the IF is executed. It is important to note that the first period after the IF ends that IF statement.

EXAMPLE:

```
IF A IS LESS THAN 0
  ADD 1 TO X
  ADD 2 TO Y.
  ADD 3 TO Z.
```

```
IF A IS LESS THAN 0
  ADD 1 TO X.
  ADD 2 TO Y
  ADD 3 TO Z.
```

NOTE: Army naming convention is not used for this example for illustrative purposes.

In the first case, if A is less than zero, one is added to X and two is added to Y then three is added to Z. If A is not less than zero, the adds to X and Y are skipped, then three is added to Z. The period ends the control of the IF. When the condition after the IF is true, the statements under control of the IF (e.g., the statements between the condition and the period) are executed. When the condition is not true, the statements under control of the IF are skipped and execution begins with the first statement after the period.

In the second case, if A is less than zero, one is added to X, the two is added to Y, and three is added to Z. If A is not less than zero, the add to X is skipped, then two is added to Y and three is added to Z.

Periods are very important in the logical construction of an IF statement. The omission or addition of a period can radically effect the execution of a program.

EXAMPLE:

```
IF A IS LESS THAN B
  ADD 1 TO A-COUNT
  GO TO A-EXIT.
ADD 1 to B-COUNT.
```

215

or


```

IF A IS LESS THAN B
  ADD 1 TO A-COUNT.
  GO TO A-EXIT.
ADD 1 TO B-COUNT.

```

NOTE: Army naming conventions are not used in this example for illustrative purposes.

In the first case, the branch to A-EXIT is only taken if A is less than B. If A is not less than B, then the statement after the period ending the IF is executed (ADD 1 TO B-COUNT). In the second case, if A is less than B or not less than B, the branch to A-EXIT is executed since the period ending the IF statement is after the ADD 1 TO A-COUNT. B-COUNT will never be incremented since execution cannot reach that point.

Another form of the IF statement is:

IF condition any series of COBOL verbs with associated parts (no periods)

ELSE any series of COBOL verbs with associated parts.

EXAMPLE:

```

IF IMØ1-DIVISION NOT = WS-DIVISION-SAVE
  OR WS-LINE-COUNT GREATER THAN 45
  PERFORM Ø15Ø-HEADING-ROUTINE THRU Ø15Ø-HEADING-EXIT
  MOVE 1 TO WS-LINE-COUNT
ELSE
  ADD 1 TO WS-LINE-COUNT.
  MOVE IMØ1-NAME TO WS-NAME.

```

In the above example, the first condition was made up of two parts (a compound condition) consisting of IMØ1-DIVISION NOT = WS-DIVISION-SAVE and WS-LINE-COUNT GREATER THAN 45. The OR between the two tests means that if either (or both) of the tests is true, then everything until the ELSE is to be executed in order. If both of the tests are false, then everything after the ELSE will be executed instead. Then control is given to the sentence after the period (a MOVE statement).

Another connector between tests is the word AND. AND indicates that both conditions have to be true for the statements under the IF to be executed. If they are not true (either or both) then execution is given to the statements after the ELSE if present. If there is no ELSE associated with the IF, then execution goes to the first statement after the period ending the IF.

EXAMPLE:

```

IF IMØ1-DIVISION = WS-DIVISION-SAVE
    AND IMØ1-SOC-SEC-NUM EQUAL TO IT1Ø1-SOC-SEC-NUM
    PERFORM Ø1Ø1-UPDATE-ROUTINE THRU Ø1Ø1-UPDAIE-EXIT
ELSE IF IMØ1-DIVISION = WS-DIVISION-SAVE
    AND IMØ1-SOC-SEC-NUM GREATER THAN IT2Ø1-SOC-SEC-NUM
    PERFORM Ø1Ø2-ADD-ROUTINE THRU Ø1Ø2-ADD-EXIT.
MOVE IMØ1-NAME TO WS-NAME.

```

In the above example, the first IF has a compound condition consisting of whether IMØ1-DIVISION was EQUAL TO WS-DIVISION-SAVE AND also whether IMØ1-SOC-SEC-NUM was EQUAL TO IT1Ø1-SOC-SEC-NUM. If both tests were true, then Ø1Ø1-UPDATE-ROUTINE is PERFORMed. If either or both tests were false, then the statements after the ELSE are executed. The second IF also is made up of two tests: whether IMØ1-DIVISION is EQUAL TO WS-DIVISION-SAVE AND whether IMØ1-SOC-SEC-NUM is GREATER THAN IT2Ø1-SOC-SEC-NUM. If both tests were true, then Ø1Ø2-ADD-ROUTINE is PERFORMed. If either or both of the tests were false, then the MOVE statement after the period ending the entire IF is executed. Notice that everything from the first IF to the final period is all one sentence (one logical construction). If any period had been in the middle of the sentence, then the IF will have ended there. Periods are very important things in IF statements. Notice also that one can (and many times should) put another IF as one of the COBOL verbs whose execution depends on the trueness or falseness of the original IF. This is called nesting IF's. In the above example, the test for equality between OMØ1-DIVISION and WS-DIVISION-SAVE is made twice. An example of how to avoid this and still have the same logical construction is given below.

EXAMPLE:

```

IF IMØ1-DIVISION = WS-DIVISION-SAVE
    IF IMØ1-SOC-SEC-NUM = IT1Ø1-SOC-SEC-NUM
        PERFORM Ø1Ø1-UPDATE-ROUTINE THRU Ø1Ø1-UPDAIE-EXIT
    ELSE
        IF IMØ1-SOC-SEC-NUM GREATER THAN IT2Ø1-SOC-SEC-NUM
            PERFORM Ø1Ø2-ADD-ROUTINE THRU Ø1Ø2-ADD-EXIT.
MOVE IMØ1-NAME TO WS-NAME.

```

In the above example, the first IF has only one test: whether IMØ1-DIVISION is EQUAL TO WS-DIVISION-SAVE. Everything else in the sentence is dependent on the trueness or falseness of that test. If it was false, then control is given to the MOVE statement after the period. If it was true, then the second IF is executed. If the second condition was true, then Ø1Ø1-UPDATE-ROUTINE is PERFORMed. After Ø1Ø1-UPDATE-ROUTINE is finished, control is given to the MOVE statement after the period. If the second condition is false, then the third IF is executed. If that condition is true, then Ø1Ø2-ADD-ROUTINE is PERFORMed. After Ø1Ø2-ADD-ROUTINE is finished, control is given to the MOVE statement after the period. Notice that the first and third IF's have no ELSE associated with them. The rule for matching IF's with ELSE's is that an ELSE is associated with the first possible IF above the ELSE.

This allows the programmer to build very complex logical constructions with compound conditions and nested IF's. Tremendous care should be exercised when designing these constructions.

If necessary, parentheses should be used in compound conditions to assure evaluation in the proper order. If OR and AND are used in the same compound, then the AND test is evaluated first.

EXAMPLE:

```
IF A IS NOT GREATER THAN B
  OR A + B IS EQUAL TO C
  AND D IS POSITIVE
```

will be evaluated as though it were parenthesized as follows:

```
IF (A IS NOT GREATER THAN B)
  OR (((A + B) IS EQUAL TO C)
    AND (D IS POSITIVE))
```

NOTE: Army naming conventions are not used for these examples for illustrative purposes.

The use of parentheses never will hurt a program's efficiency and can only add to the clarity of what the programmer is trying to do. Also when going from one computer to another manufacturer's computer, the evaluation rules could conceivably differ. With parentheses, there is no doubt for either the compiler or another programmer having to make changes to the program in the absence of the original programmer.

If at some point inside the nested IF's the programmer would like to simply exit from the entire construction and go to the sentence after the period, the COBOL reserved phrase NEXT SENTENCE may be used.

EXAMPLE:

```
IF condition-1
  IF condition-2
    IF condition-3
      NEXT SENTENCE
    ELSE
      IF condition-4
        .
      ELSE NEXT SENTENCE
  ELSE
    IF condition-5
      .
    .
```

208

ELSE

.

ELSE

NEXT SENTENCE.

56A

219

In the above example, if conditions 1, 2, and 3 are true, then control is simply given to the sentence after the period (the periods inside the nested IF's in the example only indicate a series of COBOL verbs and associated parts). If conditions 1 and 2 are true and conditions 3 and 4 are false, then again control is simply passed to the sentence after the period. If condition 1 is false, then the associated ELSE means to execute the sentence after the period. The last ELSE NEXT SENTENCE (period) is superfluous and was included to show the point. If the ELSE was not there and condition 1 was false, control would be given to the sentence after the period anyway. Some programmers consider it good practice to match all IF verbs with an ELSE even if the ELSE is superfluous as in this case. If NEXT SENTENCE is put immediately after an ELSE or the conditions following an IF, no other verb except ELSE may follow it. Putting NEXT SENTENCE after a series of verbs and associated parts simply tells the compiler to do what it would have done anyway.

EXAMPLE:

```
IF condition-1
.
.
NEXT SENTENCE
ELSE
.
.
.
```

In the above example, NEXT SENTENCE is unnecessary. Control will go to the next sentence anyway.

88 level items can be used in conjunction with an IF.

EXAMPLE:

```
03 WS-PAY PIC 9(6)V99.
88 SMALL VALUE .00 THRU 100.00.
.
.
.
IF SMALL
PERFORM 0505-SMALL-PAY-PROCESSING THRU 0505-SMALL-EXIT
ELSE
.
.
.
```

is equivalent to:

```
IF (WS-PAY EQUAL TO .00 OR WS-PAY GREATER THAN .00)
AND (WS-PAY LESS THAN 100.00 OR WS-PAY EQUAL TO 100.00)
PERFORM 0505-SMALL-PAY-PROCESSING THRU 0505-SMALL-EXIT
ELSE
```

Clearly it is more convenient to use the 88 level item and there is the added benefit of being able to change the ranges of things being tested for by simply changing one card. 88's also increase the readability of the program and are self documenting with properly chosen names.

Army Standard:

- (1) Use of the nested IF other than by the original programmer is prohibited.
- (2) Compound conditions requiring use of both AND and OR are prohibited.
- (3) NOT will not be used in compound conditions.
- (4) Avoid:
 - (a) Using the NUMERIC and ALPHABETIC test unnecessarily.
 - (b) Comparing numeric items with different number of decimal places. Move one of the items to a holding field that has the same PICTURE instead.
 - (c) Comparing non-numeric items with different size PICTURES.
 - (d) Comparing group items that have unused areas in them.
 - (e) Using the phrase ELSE NEXT SENTENCE unless it is in a nested IF.
- (5) Do not specify a GO TO in and IF that goes to some code then returns the statement after the IF. Use a PERFORM instead or incorporate the code into the IF itself.
- (6) NEXT SENTENCE can often be avoided by reversing the condition in the IF (using a NOT).
- (7) Don't test the same data item for the same value once you have determined what the value is.
- (8) Try to test for ranges of values instead of specific values, when appropriate.
- (9) Don't use the ALPHABETIC test. Use IF...NOT NUMERIC instead.

EXAMPLES:

	More efficient	Less efficient
#1	IF A EQUALS B MOVE X TO Y PERFORM Z THRU Z-EXIT. MOVE	IF A EQUALS B GO TO C. D. MOVE C. MOVE X TO Y. PERFORM Z THRU Z-EXIT. GO TO D.
#2	IF A NOT EQUAL TO B MOVE C TO D.	IF A EQUALS B NEXT SENTENCE ELSE MOVE C TO D.
#3	IF A EQUALS 1 MOVE X TO B ELSE IF A EQUALS 7 MOVE Y TO B ELSE IF A EQUALS 9 MOVE Z TO B.	IF A EQUALS 1 MOVE X TO B. IF A EQUALS 7 MOVE Y TO B. IF A EQUALS 9 MOVE Z TO B.

#4

```

IF A LESS THAN 5
  IF A LESS THAN 1
    PERFORM C THRU C-EXIT
  ELSE
    PERFORM B THRU B-EXIT
ELSE
  PERFORM C THRU C-EXIT

```

211

```

IF A EQUALS 1
  PERFORM B THRU B-EXIT.
IF A EQUALS 2
  PERFORM B THRU B-EXIT.
IF A EQUALS 3
  PERFORM B THRU B-EXIT.
IF A EQUALS 4
  PERFORM B THRU B-EXIT.
IF A LESS THAN 1
  PERFORM C THRU C-EXIT.
IF A GREATER THAN 4
  PERFORM C THRU C-EXIT.

```

PERFORM

The PERFORM statement has a number of forms each having its own application. They will be discussed in detail one at a time. The basic concept of a PERFORM is common for all of them. While writing a program, that program can always be broken up into several discrete parts which function independently of each other. The main control section of the program simply gives control to the appropriate part at the appropriate time through a series of PERFORM statements. When the part that has been given control is finished, control is given to next COBOL verb after the PERFORM statement.

EXAMPLE:

```

PERFORM 0100-ADD-ROUTINE THRU 0100-ADD-EXIT.
CLOSE ADD-FILE.

```

```

0100-ADD-RTN.
  MOVE IT01-ADD-RCD TO OM01-NEW-MASTER-RECORD.
  WRITE OM01-NEW-MASTER-RECORD.
0100-ADD-EXIT. EXIT.
0200-SUBTRACT-RTN.

```

In the above example, when control is passed to the PERFORM 0100-ADD-RTN THRU 0100-ADD-EXIT statement, the PERFORM statement passes control to 0100-ADD-RTN. IT01-ADD-RCD is moved to OM01-NEW-MASTER-RECORD and then OM01-NEW-MASTER-RECORD is written. As soon as the 0100-ADD-EXIT paragraph is executed, the COBOL verb EXIT tells the machine to transfer control back to the PERFORM which passed control to thee 0100-ADD-RTN. The PERFORM then passes control to the next COBOL verb after the PERFORM (CLOSE ADD-FILE). The PERFORM is really a 'go do this then come back when your finished' verb. A simpler version, but less preferable, is:

212
EXAMPLE:

PERFORM 0100-ADD-RTN.
CLOSE ADD-FILE

0100-ADD-RTN.
MOVE ADD-RECORD TO NEW-MASTER-RECORD.
WRITE NEW-MASTER-RECORD.
0020-SUBTRACT-RTN.

Army Standard: An EXIT paragraph should always be the last paragraph in a PERFORM statement.

In the above example, exactly the same thing happens as the last example. While before the EXIT was explicitly defined in the PERFORM statement and written at the end of the paragraph that was being PERFORMED, this time the EXIT is implied to be at the end of the ADD-RTN paragraph just before the next paragraph name (0200-SUBTRACT-RTN.) and after the last period at the end of the last sentence in ADD-RTN (WRITE NEW-MASTER-RECORD.). It is good programming procedure to always use an explicit EXIT and PERFORM a paragraph through that EXIT. This makes the program more readable and makes the job of a maintenance programmer fixing the program at a later date much easier. The reason for this will become clearer during the discussion of the GO TO verb.

Another form of the PERFORM verb is:

PERFORM paragraph-name [THRU paragraph-exit]
UNTIL condition.

This form will PERFORM the paragraph as many times as necessary to make the condition true. This form can make an indefinite loop if the condition is never true. An example of this would be if the condition was $A = B$ AND $A \text{ NOT } = B$. This can never be true so the paragraph will be executed forever or until the operator gets tired of looking at the program executing. Clearly, this is not a desirable thing to do. Care should be used to insure that the condition will be made true at some point in the processing. If the condition is true when the PERFORM is executed, then the paragraph is never executed at all and control passes to the next verb. The condition may be an 88 level item. See the IF statement discussion for what it means.

Army Standard: (1) Always use PERFORM...THRU.
(2) Always execute the EXIT paragraph to return to the PERFORM.

EXAMPLE:

01 WS-EUF-TABS.
03 MASTER-EUF-SWITCH PIC X VALUE '0'.
88 MASTER-DONE VALUE '1'.

PERFORM 0010-PARA-1 THRU 0010-PARA-EXIT
UNTIL MASTER-DONE.

60 223

This example will PERFORM 0010-PARA-1 THRU 0010-PARA-EXIT until '1' is moved to WS-MASTER-EOF-SWITCH. When this happens, the 88 under WS-MASTER-EOF-SWITCH becomes true and the UNTIL is satisfied.

PERFORM paragraph-name [THRU paragraph-exit]
n TIMES.

This form of the PERFORM verb will PERFORM the paragraph a fixed number of times where n is an integer that specifies how many times the paragraph is to be executed. If n is a variable, then it must be an integer.

EXAMPLE:

```
77 PERFORM-COUNT                PIC S9(4)
      .                          USAGE IS COMP
      .                          VALUE IS +20
      .                          SYNC.
PERFORM 5001-PARA-1 THRU 5001-PARA-EXIT
      PERFORM-COUNT TIMES.
```

or

```
PERFORM 5001-PARA-1 THRU 5001-PARA-EXIT
      20 TIMES.
```

Both of the examples do the same thing. They will PERFORM 5001-PARA-1 through 5001-PARA-EXIT 20 times.

The last form of the PERFORM verb is:

PERFORM paragraph-name [THRU paragraph-exit]
VARYING name-1 FROM value-1 BY value-2
UNTIL condition.

This form will initialize name-1 (index-name-1 or numeric-data-item-1) to value-1 (integer-1, index-name-2, or numeric-data-item-2) and test the condition (a simple or compound condition). If the condition is false, the paragraph is PERFORMed. When the paragraph is done the first time, name-1 is incremented by value-2 (integer-2, index-name-3, or numeric-data-item-3) and the condition is tested again. This process of PERFORMing the paragraph, incrementing name-1 and testing the condition continues until the condition is true. If the condition is true when the PERFORM is executed the first time, then control immediately passes to the next verb and the paragraph is never performed.

EXAMPLE:

```
PERFORM 0100-PARA-1 THRU 0100-PARA-EXIT
      VARYING TABLE-INDEX FROM 1 BY 1
      UNTIL TABLE-INDEX IS GREATER THAN 25.
```

This example initializes TABLE-INDEX to 1, then tests whether TABLE-INDEX is greater than 25. If it is not so, 0100-PARA-1 is executed. When control finally reaches 0100-PARA-EXIT, TABLE-INDEX is incremented by 1 (giving 2) and the condition is again evaluated. TABLE-INDEX is still not greater than 25, so 0100-PARA-1 again gets control; however, TABLE-INDEX is now equal to 2. This type of PERFORM is very useful in searching tables for an entry. More of how to use this form will be discussed in the section on tables.

EXIT

EXIT is a verb that provides a way to terminate a paragraph being PERFORMed and then return to the PERFORM statement for further processing if necessary. It can be the only thing after the paragraph-name other than comments ('*' in column 7) until the next paragraph-name.

GO TO

The GO TO is the most abused single verb in COBOL. When an error is found in a program, the average programmer immediately begins to think in terms of GOing somewhere else. This leads to many unpredictable results initially and usually has some hidden implications that may not show up until the next programmer has to repair the program again. A study was done to try to determine the cause of COBOL logical errors. The result of this study was to find the GO TO guilty of the vast majority. In this context, this text will present the way a GO TO may be used and then place several large constraints on how to use it. The form of the GO TO is:

GO TO paragraph-name.

EXAMPLE:

GO TO 0100-PARA-1.

In this example, the sentence immediately following 0100-PARA-1 will be executed. Indiscriminate use of this type of statement can lead to 'spaghetti' logic that is hard to debug and repair at a later date. A much more restrictive use is:

EXAMPLE:

GO TO 0100-PARA-EXIT.

This is the only type of GO TO allowed in 'structured' programming. In a given paragraph, the programmer can only go to the EXIT of that paragraph.

EXAMPLE:

```

Ø525-PARA-1.
  IF LINE-COUNT GREATER THAN 2Ø
    GO TO Ø525-PARA-EXIT.

```

```

Ø525-PARA-EXIT. EXIT.

```

This type of limitation requires that each paragraph be entered via a PERFORM statement. Most paragraphs won't need a GO TO in them at all. However, at some future date, another programmer may find it necessary to put one in. To keep the program compatible, the initial programmer should always have an EXIT for each paragraph and PERFORM THRU that exit. If the PROGRAM was filled with a mixture of PERFORMs and PERFORM THRUs and suddenly another programmer needed to change one of the PERFORMs to a PERFORM THRU, all of the PERFORMs for that paragraph would have to be changed, leading to a lot of work and possible source of new problems. If all of the paragraphs have EXITS, then the programmer simply GO TOs the EXIT and makes no other changes to the program. This cuts down on maintenance time, a major cause of cost overrun on a project and also makes the original programmer spend more time on the design of the program leading to more trouble free code. Some programmers complain that COBOL is already too verbose and wordy and all of these restrictions make his job harder and longer. That may be true, but the real cost of a program is not in the writing of it; it is in the maintenance of it. By forcing the programmer to conform to this type of design forces him to use routine (break the program up into pieces that do discrete, separate things). This makes the problem clearer and the logic simpler.

STOP

This verb is used to either temporarily halt a program or to terminate it altogether. The form of the STOP statement is:

```

STOP   RUN
        quoted-literal

```

EXAMPLES:

```

STOP RUN.
STOP 'MOUNT 24Ø2-TEST-DATA-TAPE'

```

The first example terminates the program. The second example halts the program and types the literal on the console typewriter. The operator will then restart the program after appropriate action has been taken on the message typed.

- Army Standard:
- (1) There should be only one STOP RUN in the program. This should be a single statement paragraph.
 - (2) All files must be closed before the STOP RUN is executed.

ON (IBM EXTENSION)

The ON verb allows the programmer to specify when a series of COBOL verbs are to be executed. Its form is:

ON integer-1 [AND EVERY integer-2] [UNTIL integer-3]
 any series of COBOL verbs and associated parts except IF-ELSE
 NEXT SENTENCE
 { ELSE any series of COBOL verbs including IF-ELSE }
 { OTHERWISE NEXT SENTENCE } .

This verb sets up a count-condition. If only integer-1 has been specified, then the count-condition will only be satisfied once. The count is incremented by one each time the ON statement is executed. If the count-condition is not satisfied, the series of COBOL verbs after the ELSE is executed. If no ELSE is specified and the count-condition is not satisfied, execution goes to the next sentence after the period terminating the ON statement.

Army Standard: ON should only be used during debugging.

EXAMPLE:

```
ON 1 MOVE ZEROS TO WS-LINE-COUNT
    PERFORM 5626-HEADING-ROUTINE THRU 5626-HEADING-EXIT.
```

```
ON 10 AND EVERY 10
    PERFORM 1000-SUB-TOTAL-ROUTINE THRU 1000-SUB-EXIT.
```

```
ON 1 UNTIL 1000
    READY TRACE
ELSE RESET TRACE.
```

```
ON 25 AND EVERY 25 UNTIL 975
    NEXT SENTENCE
ELSE PERFORM 0500-PROCESS-ROUTINE THRU 0500-PROCESS-EXIT
    PERFORM 0600-COUNT-ROUTINE THRU 0600-COUNT-EXIT.
```

The first example executed the MOVE and PERFORM the first time the statement is executed. The second example will execute the PERFORM every ten times the statement is executed. The third example initiates a TRACE (described next) the first 1000 times the statement is executed. After the 1000th time, the ELSE path is taken and the TRACE is terminated. The fourth example simply goes to the period after the statement on the 25th time and each subsequent 25th time until the ON statement has been executed 975 times. If the particular time the ON is being executed is not one of the 25th times or is past the 975 limit, then the two PERFORMs under the ELSE are executed. This statement works very similarly to the IF-ELSE statement where the condition is described by the ON.

DEBUGGING

Tracing is a powerful tool for the programmer. The format of the TRACE statement is:

{READY} TRACE (IBM EXTENSION)
{RESET}

EXAMPLES:

READY TRACE.
RESET TRACE.

When the first example is executed, the COBOL listing number (number to the extreme left on the listing) of the paragraphs will be listed on the printer as they are executed. Any lines the program writes on the printer will be interspersed at the appropriate places. This allows the programmer to actually see the order that the paragraphs in his program were executed and to hopefully find out why the logic is going awry. The second example turns off the trace. If the trace is off when it is turned off again, nothing happens. Likewise if the trace is on when it is turned on again, nothing happens. The TRACE verb can generate a large amount of paper and should be used with discretion. Intermediate values can be printed using the DISPLAY verb or the EXHIBIT verb during the run while using the trace and a clearer picture of what is going on is made available. A powerful use of the TRACE verb is in conjunction with an IF.

Army Standard: READY TRACE and RESET TRACE is only to be used as a debugging aid.

EXAMPLE:

IF IMØ1-ACCOUNT-NUMBER = 9546251
READY TRACE.

This is used to TRACE beginning at a problem account number. Presumably processing up to this point has gone correctly and no TRACE was necessary. This saves paper and processing time.

TABLE HANDLING

A table is defined in the DATA DIVISION via the OCCURS clause. It consists of a series of elements identical in form.

is defined by:

PIC X(20)
OCCURS 25 TIMES.

EXAMPLE:

MOVE WS-NAME (1) TO WS-DETAIL-NAME.

EXAMPLE:

•

```
MOVE 1 TO WS-SUB.  
MOVE WS-NAME (WS-SUB) TO WS-DETAIL-NAME.
```

DRAFT ST 18-150

EXAMPLE (LINEAR STYLE):

```

MOVE 1 TO WS-SUB.
MOVE '0' TO WS-FLAG.
A. IF IM01-NAME EQUAL TO WS-NAME (WS-SUB)
    MOVE '1' TO WS-FLAG
    GO TO B.
ADD 1 TO WS-SUB.
IF WS-SUB LESS THAN 25
    GO TO A.
B. ADD 1 TO WS-CARD-COUNT.

```

This example initializes WS-SUB to 1 then searches the name table for a match with IM01-NAME. If a match is found, WS-FLAG is set to 1. If a match is not found after all the names have been examined, WS-FLAG is set to 0. Another way to do the same thing is to use the PERFORM VARYING verb.

EXAMPLE (STRUCTURED STYLE):

```

MOVE '0' TO WS-FLAG
PERFORM 0100-NAME-COMPARE THRU 0100-NAME-EXIT
    VARYING WS-SUB FROM 1 BY 1
    UNTIL WS-SUB IS GREATER THAN 25
        OR WS-FLAG = '1'
ADD 1 TO WS-CARD-COUNT.

```

```

0100-NAME-COMPARE.
    IF IM01-NAME = WS-NAME (WS-SUB)
        MOVE '1' TO WS-FLAG.
0100-NAME-EXIT. EXIT.

```

This example sets WS-FLAG to '0' then PERFORMS 0100-NAME-COMPARE through 0100-NAME-EXIT. During the process of the PERFORM, WS-SUB is set to 1 and the conditions are tested. If the conditions are not true, then 0100-NAME-COMPARE through 0100-NAME-EXIT are executed. After 0100-NAME-EXIT is finished, WS-SUB is incremented BY 1 and the conditions are tested. As soon as one of the conditions are true, the PERFORM statement is finished and the ADD 1 TO WS-CARD-COUNT is executed.

If the table consists of multiple entry items (e.g., name and social security number), the elementary items are referenced as though they are subscripted.

EXAMPLE:

```

Ø1 WS-NAME-SSN-TABLE.
  Ø5 WS-NAME-SSN                OCCURS 25 TIMES.
    1Ø WS-NAME                  PIC X(2Ø).
    1Ø WS-SSN                   PIC X(9).
.
.
.
IF WS-NAME (WS-SUB) EQUAL TO IMØ1-NAME
  MOVE '1' TO WS-FLAG.

```

Even though the description of WS-NAME does not contain an OCCURS clause, WS-NAME is subordinate to WS-NAME-SSN which does contain an OCCURS clause. This being the case, WS-NAME must be referenced as though WS-NAME itself contained the OCCURS. Another optional entry with the OCCURS clause is INDEXED BY. This clause specifies an index to be used with the particular table defined by the rest of the description.

EXAMPLE:

```

Ø1 WS-BAS-TABLE.
  Ø5 WS-BAS-ENTRY                OCCURS 3Ø TIMES
                                INDEXED BY BAS-INDEX.
    1Ø WS-BAS                   PIC S999V99.
    1Ø WS-RANK                  PIC X(3).

```

BAS-INDEX is the index associated with the BAS-ENTRY table. WS-BAS and WS-RANK are referenced exactly as in using a subscript.

EXAMPLE:

MOVE WS-BAS (BAS-INDEX) TO WS-DETAIL-BAS.

To initialize an index to a value and decrement or increment, the index, the SET verb, is used.

EXAMPLES:

```

SET BAS-INDEX TO 1.
SET BAS-INDEX UP BY 2.
SET BAS-INDEX DOWN BY 1.

```

Indexes are used in conditions exactly like subscripts.

EXAMPLE:

IF BAS-INDEX GREATER THAN 25

.
.
.

The linear example given at the beginning of the table handling section would look like this, using indexing instead of subscription, assuming that INDEXED BY NAME-INDEX is specified after the OCCURS clause.

EXAMPLE:

```

SET NAME-INDEX TO 1.
MOVE '0' TO WS-FLAG.
A. IF IM01-NAME EQUAL TO WS-NAME (NAME-INDEX)
    MOVE '1' TO WS-FLAG
    GO TO B.
SET NAME-INDEX UP BY 1.
IF NAME-INDEX LESS THAN 25
    GO TO A.
B. ADD 1 TO WS-CARD-COUNT.

```

The structured example would look like:

```

MOVE '0' TO WS-FLAG.
PERFORM 0100-NAME-COMPARE THRU 0100-NAME-EXIT
    VARYING NAME-INDEX FROM 1 BY 1
    UNTIL NAME-INDEX IS GREATER THAN 25
        OR WS-FLAG = '1'.
ADD 1 TO WS-CARD-COUNT.
.
.
.
0100-NAME-COMPARE.
    IF IM01-NAME EQUAL TO WS-NAME (NAME-INDEX)
        MOVE '1' TO WS-FLAG.
0100-NAME-EXIT. EXIT.

```

Indexing is usually an order of magnitude more efficient than subscripting when stepping through a table from beginning to end. When using an index in a PERFORM VARYING statement, all of the SET statements are done automatically by the PERFORM.

88 entries cannot be used in or under an OCCURS.

- Army Standard:
- (1) Indexing is more efficient than subscripting and is preferred.
 - (2) Checks should be made to insure that indexes and subscripts do not exceed the range of the OCCURS.
 - (3) Subscripting or indexing must be limited to three levels.
 - (4) Subscripting or indexing beyond one level should be avoided.

DOS JOB CONTROL LANGUAGE (JCL)

// JOB name

This is the first card of each job. Name must be from 1 to 7 characters, the first of which must be alphabetic, the last of which may be characters or numbers. // is punched in columns 1 and 2.

// OPTION option-1, option-2, option-3...

This may appear after the JOB card and is used to change any of the standard system options. It is not required if the options which are standard are the options desired.

Options include:

LINK	which specifies that the job is to be written on disk for link editing.
LIST	which specifies that a listing of a compilation is to be written on SYSLST.
ERRS	specifies that a listing of any errors, if any, are to be listed on SYSLST.
DUMP	specifies that a hexadecimal dump of memory is to be made if the job is terminated abnormally.
LISTX	specifies that an ALC listing of generated instructions is to be written on SYSLST.
SYM	specifies that a table of data-names are to be listed with their corresponding system-assigned names with locations.
LOG	specifies that a listing of all job control is to be written on SYSLST.
XREF	specifies that a cross reference dictionary of data-names be written on SYSLST.
DECK	specifies that the compiler is to punch an object deck on SYSPCH.

The standard system options at the Institute of Administration are LIST, ERRS, NOLINK, NODUMP, NOLISTX, NOSYM, NOLOG, NOXREF, NODECK. Note that any option may be negated by preceding it with NO. If the programmer wanted to change one of the above options, such as NOLINK to LINK, he would punch the following card:

// OPTION LINK

If he wanted to change 2 items, such as NOLINK to LINK and LIST to NOLIST,

// OPTION LINK,NOLIST

NOTE: No space is permitted between a comma and the next option.

```
// ASSGN SYSnnn,X'cuu'
```

The assign clause is used to assign a system logical number to a specific hardware device.
nnn represents a 3 digit system logical number; cuu represents a channel and unit number to which a device is connected.

Devices include:

card reader	X'00C'
card punch	X'00D'
printer	X'00E'
tape drives	X'180' X'181' X'182' X'183' X'184' X'185'
disk drives	X'130' X'131' X'132' X'133' X'134' X'135' X'136'
display (video)	X'020' X'021'
console typewriter	X'01F'

System logical numbers on our system are available from 000 thru 029.

```
// EXEC program-name
```

This statement causes a program on core-image library to be read into memory and executed. If no name is specified, the program compiled will be read into memory and executed.

EXAMPLE:

```
// EXEC LNKEDT
```

would cause the link editor to be read into memory and link edit the program just compiled.

```
// EXEC
```

would cause the program just link edited to be read into memory and executed.

```
/*
```

/* indicates end-of-data. It would appear after the last card of the source program to be compiled, for to the compiler the source program is data. It would also appear after any data which may have been included for use by an object program.

```
/&
```

/& indicates end-of-job. When this card is encountered, the job is terminated. The JOB card of the next job should follow a /& card.

224

EXAMPLE of a job to be compiled:

```
// JOB SAMPLE
// ASSGN SYS004,X'132'
// EXEC FCOBOL
    IDENTIFICATION DIVISION.
        PROGRAM-ID. PE1.
        etc.
    rest of source program
/*
/ &
```

NOTE: The full (ANSI) COBOL compiler requires a work file on disk, and expects SYS004 to be assigned to a disk. Our disk file used for this purpose is X'132', and thus the ASSGN card shown above is required to compile ANSI COBOL programs.

EXAMPLE of a job to be compiled and executed with data included:

```
// JOB EXAMPLE
// OPTION LINK
// ASSGN SYS004,X'132'
// EXEC FCOBOL
Source deck
/*
// EXEC LNKEDT
// ASSGN SYS005,X'00C'
// ASSGN SYS006,X'00E'
// EXEC
Card data
/*
/ &
```

NOTE: Note that SYS005 must be assigned to a card reader in the COBOL source program since JCL has specified that SYS005 is to be assigned to X'00C', a card reader. Likewise, SYS006 must be assigned to a printer in the COBOL source program.

235

OS JOB CONTROL LANGUAGE (JCL)

There are three kinds of cards normally used in a typical OS job deck: the JOB card, the EXEC card, and the DD card. They will each be described to give an appreciation of the relationship to a COBOL program.

The JOB card has the following format:

```
//jobname JOB ... (other parameters)
```

This is always the first card in all JCL decks. It tells the operating system that a job is beginning.

The EXEC card has the following format when used in the context of a COBOL program being compiled, link-edited, and executed:

```
//      EXEC COBUCLG,PARM.COB=(compiler options),
//      PARM.LKED=(linkage-editor options),
//      COND.LKED=(n,LT),COND.GO=(n,LT)
```

The three cards shown, each beginning with a //, are treated as one card because the second and third cards are merely continuations of the first. A continuation is indicated by the card being continued ending with a comma and the next card beginning with a // followed by at least one space then the continued statement. In this case, the EXEC card is invoking a three step procedure (or PROC) that compiles a COBOL program, link-edits it, then executes that program. COBULCG is the name of the procedure. The first step in the procedure is named the COB step. PARM.COB refers to options that affect the execution of this step. Some of the commonly used options are:

QUOTE indicates that a double quote is used to delineate literals instead of
APOST the apostrophe (or single quote) or vice versa.

SPACE 1

SPACE 2

SPACE 3 indicate that the source program listing is to be single, double, or triple spaced respectively.

XREF indicates that a cross reference is to be generated or suppressed of
NOXREF all of the data-names and paragraph-names after the source program listing.

PMAP indicates that a pseudo-assembler listing is to be printed or
NOPMAP suppressed of the generated code in the PROCEDURE DIVISION after the source program listing and xref listing (if included).

DMAP indicates that a map of the DATA DIVISION is to be printed or
NODMAP suppressed after the source program listing.

If only one of the options is chosen, then the parentheses are not needed. If more than one option is chosen, the parentheses must be included and the options must be separated by commas.

The second step in the procedure is the LKED step. This is the step that link-edits the program. During this step, any subroutines that are used by the COBOL program are joined to it and a lot of busy work is done. There are two commonly used options in the linkage-editor.

MAP indicates that a map of the linked program is to be printed. This is useful in debugging the program using a core dump.

XREF indicates that a cross reference of the communication between the programs and subroutines is to be printed along with a map of the module.

The COND.LKED parameter is used to force execution of the LKED step even though diagnostics (errors) above the level of W have been generated during the compilation of the COBOL program. Normally a program will execute with C level errors but not E level or higher. To allow the program to link and execute with a C level error, substitute an 8 for the n in this parameter and in the COND.GO parameter. If linking and k execution is desired only with W level errors, then omit both COND parameters and the comma after the parentheses on the last PARM parameter.

After the EXEC card is the SYSIN card for the COBOL program. Its format is:

```
//COB.SYSIN DD *
```

COB indicates that it is to be used in the COB step; SYSIN is the name of the DD card; DD indicates that it is indeed a DD card (data description); and * indicates that this is a card file with the cards immediately following the DD card. The COBOL program is placed immediately behind this card.

After the last card in the COBOL program, the DD cards that describe the files used in the COBOL program. Their format is:

```
//GO.ddname DD ... (additional parameters)
```

The GO indicates that it is used during the GO step (execution); the ddname is identical to the external name used in the system name after the ASSIGN in the FILE-CONTROL paragraph in the COBOL program; the DD indicates that this is a DD card; and the additional parameters can be many things. If this was to be a card file, then an asterisk (*) is all that is necessary after the DD. If this was a card punch file then SYSOUT=B is needed after the DD. If this was a printer file, then SYSOUT=A is needed after the DD. If the only entry after the DD is the word DUMMY and the file is used for sequential output (e.g., as a printer), everything written in that file is ignored by the system. If the DUMMY file is used as sequential input (e.g., as a card reader), the first read that is executed generates an end of file condition on that file. Random reads or writes are not allowed to DUMMY files.

If DISPLAY, EXHIBIT, or TRACE is used in the program, then the following card is needed.

```
//GO.SYSOUT DD SYSOUT=A
```

EXAMPLE:

```
//EXAMINE JOB ... (other JOB card parameters)
//      EXEC COBUCLG,
//      PARM.COB=(XREF,PMAP,DMAP,SPACE2),
//      PARM.LKED=XREF,
//      COND.LKED=(8,LT),
//      COND.GO=(8,LT)
//COB.SYSIN DD *
      IDENTIFICATION DIVISION.
```

```
      .
      .
      FILE-CONTROL.
```

```
      SELECT DISK-FILE
        ASSIGN TO UT-S-DISK.
      SELECT PRINTER
        ASSIGN TO UT-S-PRINTER.
      SELECT CARD-READER
        ASSIGN TO UT-S-READER.
      SELECT CARD-PUNCH
        ASSIGN TO UT-S-PUNCH.
```

```
      .
      . (rest of COBOL program)
```

```
//GO.DISK DD ... (other DD card parameters)
//GO.PRINTER DD SYSOUT=A
//GO.READER DD *
```

```
      . (data cards)
```

```
      .
//GO.PUNCH DD SYSOUT=B
//
```

In this example, the JOB name is EXAMPLE. The COBUCLG procedure is invoked. Parameters for the COB (COBOL compile) step are XREF, PMAP, DMAP, AND SPACE2. This will print a cross reference of all the data-names and paragraph names in the program, list all of the pseudo assembler generated codes in the procedure division, map the data division variables, and double space the COBOL source program listing. Parameters for the LKED (link-edit) step are XREF. This will map the linked module and cross reference the communication between the routines. The COND parameters indicate that linking and execution is desired with C level errors in compilation. The COB.SYSIN card indicates that the program immediately follows on cards.

The four files that the program uses are defined to have external names of DISK, PRINTER, READER, and PUNCH. These same names are all defined as DD names beneath the COBOL program. The GO.DISK DD card describes the disk file. The GO.PRINTER DD card indicates that this is a printer file by having SYSOUT=A after the DD. The GO.READER DD card indicates that this is a card file by the

228

asterisk (*) after the DD. The GO.PUNCH DD card indicates that this is a card punch file by the SYSOUT=B after the DD. Finally the last card in the deck is a // with nothing else on the card. This indicates the end of the job.

239

COBOL SOURCE DESIGN LANGUAGE (SDL)

1. ADD: Adds the contents of one data name to another data name or a numeric integer value to a data name.
2. AT END: Used only in conjunction with the READ.
3. CLOSE: Makes a file nonaccessible, the file can no longer be read from or written to.
4. EXIT: Used to identify the ending point of a routine.
5. IF-THEN-ELSE: The IF portion states a condition, the THEN portion is done when condition is true, the ELSE portion is done when the condition is not true.
6. MOVE: Moves the contents of one data name to another data name.
7. OPEN: Makes a file accessible so that it can be read from or written to.
8. PERFORM: Does a routine from its beginning point through its ending point, one time.
9. PERFORM-UNTIL: Does a routine from its beginning point through its ending point, some number of times until a stated condition is true.
10. READ: Accesses a file and makes one record available for processing.
11. STOP RUN: Used to show the end of all processing. The STOP RUN will be used at the end of the main controlling routine.
12. WRITE: Puts the contents of a record name out to a physical device.
13. NEXT SENTENCE: Used in conjunction with the IF-THEN-ELSE to show a null THEN or ELSE portion of the statement.

ANS-COBOL Programming Standards

Software Division
Computer Science Department
US Army Institute of Administration
Ft Benjamin Harrison, Indiana 46216

February 1978

IA-01-02-05

The Programming Standards established here are not intended to be all-inclusive for ANS-COBOL program development. Rather, they are presented in order to eliminate ambiguity in the completion of COBOL practical exercises and tests by students of Computer Science Department. Standards already established by US Army Computer Systems Command are included in this memorandum. It should be noted that the standards specified are mandatory for graded practical exercises and tests administered by the Software Division. Specifications are grouped by COBOL Division, i.e., Section I deals with requirements in the Identification Division, Section II with Environment Division, Section III with Data Division and Section IV with Procedure Division. Where these standards conflict with ST 18-150, these standards have precedence.

Section I. IDENTIFICATION DIVISION

Though the only entry required by the COBOL compiler is that of a Program-ID, all paragraphs must be included in every program.

Requirements:

- a. Program-ID. A meaningful name must be provided for this entry.
- b. Author. The student will enter his Rank, Name and Class Number.
- c. Installation: Enter

CSD, USAIA, Ft Harrison, IN 46216

- d. Date-Written: The date the student started coding the program will be entered here.
- e. Date-Compiled. Enter TODAY.
- f. Security. Enter UNCLASSIFIED.

IA-01-02-05

Section III. DATA DIVISION

(a) File Section.

(1) Though the only clause required by the compiler in a File Description (FD) is the LABEL RECORD clause, all optional entries will be included for all files. Each clause will be punched on a separate card and will begin in column 12. Example of an FD:

CONT	A	B	12	16	20	24	28	32	36	40	44	48
	FD		IM	PERS	-	MASTER						
			LABEL	RECORD	IS	STANDARD						
			BLOCK	CONTAINS	1	RECORDS						
			RECORD	CONTAINS	64	CHARACTERS						
			DATA	RECORD	IS	IMCL-PERS-MSTR						

The blocking factor and record size are specified on the record layouts for each PE. The data record name is a programmer-defined name and will be formed according to these standards:

(a) The first two characters will be identical to the first two characters of the file-name.

(b) The next two characters will consist of a sequence number beginning with 01. If there is only one record format associated with a file, that record name will have 01 as the third and fourth characters. If there is more than one record format, the first one described will have a sequence number of 01, the second one 02, and so on.

(c) The fifth character will always be a hyphen (-).

(d) The rest of the data record name will be descriptive of the record itself.

(e) If a record is defined as an elementary item, the picture clause will begin in column 48 (or as specified in para b(4)) and the word PICTURE will be abbreviated to PIC.

(2) Standards for defining data items within records are:

(a) Level numbers will begin at 05, and each subordinate level will be incremented by 5. Subordinate elements will be indented four (4) columns.

(b) The first five characters of all data item names will be identical to the first five characters of the record name. Allow two blank columns between level numbers and the data names.

(c) The rest of the data item name will be descriptive of the data item.

IA-01-02-05

The following is an example of the description of the record associated with
IM-PERS-MSTR, the file defined in paragraph a(1) of this section

A	B												
		12	16	20	24	28	32	36	40	44	48	52	56
01	IM01-PERS-MSTR.												
05	IM01-NAME.												
	10 IM01-LAST-NAME											PIC X(12).	
	10 IM01-FIRST-NAME											PIC X(8).	
05	IM01-SSN											PIC 9(9).	
05	IM01-RANK											PIC X(3).	
05	IM01-DATE-OF-RANK											PIC 9(6).	
05	IM01-NR-DEPS											PIC 9(2).	
05	IM01-DATE-OF-BIRTH.												
	10 IM01-DAY-OF-BIRTH											PIC 9(2).	
	10 IM01-MON-OF-BIRTH											PIC 9(2).	
	10 IM01-YR-OF-BIRTH											PIC 9(2).	
05	IM01-YTD-GROSS-PAY											PIC 9(5)N99	
	USAGE COMP-3.												
05	FILLER											PIC X(14).	

(d) The PICTURE clause for elementary items will begin in column 48 (or as specified in para b(4)). The word PICTURE will be abbreviated to PIC.

(e) Each additional clause will be punched on a separate card, and will begin in column 28.

b. Working-Storage Section.

(1) The first entry in Working-Storage will be a level 77 item indicating the beginning of the section, and the last entry will be an 01 item indicating the end of the section. Insertion of these two items aids in debugging with a core dump. (See example on page 10.)

(2) All data elements defined in WORKING-STORAGE will be preceded by characters (WS-) as the first three characters of the name.

(3) The rest of the name will be formed as follows:

(a) For level 77 and 01 items, it is sufficient to provide a descriptive name. Examples:

To name a page counter using a level 77:

CONT	A	B																			
		12	16	20	24	28	32	36	40	44	48										
77		WS	-	P	A	G	E	-	C	O	U	N	T	E	R						

To name an independent record for a printed detail line using a level 01:

CONT	A	B																			
		12	16	20	24	28	32	36	40	44	48										
01		WS	-	D	E	T	A	I	L	-	L	I	N	E							

(b) All data elements subordinate to a level 01 item will contain in positions 4 and 5 (and occasionally 6) of the name, an identifier which links them to the level 01 item: Examples: A field to be printed as part of a detail line and which contains a person's social security number could be named as: WS-HLL-SSN

A field to contain a date and be printed as part of Heading Line One could be named as: WS-HLL-DATE

(c) Rules for level numbers and indentation are the same as outlined for items in the file section. See para a(2)(a) of this section.

IA-01-02-05

(4) Any clauses included with a data item description will be in the following order:

REDEFINES
OCCURS
PIC
USAGE
VALUE
JUSTIFIED
SYNC

(5) The first clause will be punched in column 48, and each additional clause will be punched on a separate card beginning in column 28.

(6) All numeric level 77 items will be assigned an initial value.

(7) The following example of a WORKING-STORAGE SECTION is provided to clarify the standards outlined above.

Working Storage EXAMPLE

0742

CONT		A	B																
7		8	12	16	20	24	28	32	36	40	44	48	52	56	60				
		WORKING-STORAGE SECTION.																	
07		WS-BEGIN-POINT												PIC X(27)					
		VALUE 'WORKING STORAGE BEGINS HERE'.																	
07		WS-LINE-COUNTER												PIC 9(2)					
		USAGE COMP																	
		VALUE 21																	
		SYNC.																	
07		WS-IM-EOF-SWITCH												PIC X(3)					
		VALUE 'OFF'.																	
01		WS-DETAIL-LINE.																	
05		FILLER												PIC X(15)					
		VALUE SPACES.																	
05		WS-DL-NAME.																	
10		WS-DL-FIRST-NAME												PIC X(8)					
10		WS-DL-LAST-NAME												PIC X(12)					
05		FILLER												PIC X(10)					
		VALUE SPACES.																	
05		WS-DL-SSN												PIC 9(9)					
05		FILLER												PIC X(19)					
		VALUE SPACES.																	
01		WS-END-POINT												PIC X(25)					
		VALUE 'WORKING-STORAGE ENDS HERE'.																	

IA-01-02-05

Section IV. PROCEDURE DIVISION

a. PARAGRAPH Names:

(1) All paragraph names will contain in the first four positions a sequence number, beginning with 0010. The number assigned to each paragraph will be in ascending order and each will be incremented by 10 to allow insertion of additional paragraphs as the need arises. An exception will be numbers assigned to EXIT paragraphs. These will contain the same number as the preceding paragraph. All paragraphs will have an associated exit paragraph, except, of course, exit paragraphs themselves.

(2) The fifth position of the name will be a hyphen (-), and the remainder of the name will be descriptive of the functions(s) performed.

(3) Example:

CONT	A	B	12	16	20	24	28	32	36	40	44	48
			P	R	O	C	E	D	U	R	E	
			D	I	V	I	S	I	O	N	.	
			0	0	1	0	-	M	A	I	N	
			-	D	R	I	V	E	R	.		
			O	P	E	N	-	-	-	-		
			P	E	R	F	O	R	M			
			0	0	2	0	-	R	E	A	D	
			-	P	R	I	N	T				
			T	R	U							
			0	0	2	0	-	E	X	I	T	
			U	N	T	I	L	W	S	-	I	M
			-	E	O	F	-	S	W			
			E	Q	U	A	L					
			'	O	N	'	.					
			C	L	O	S	E	-	-	-	-	
			0	0	1	0	-	E	X	I	T	
			S	T	O	P		R	U	N	.	
			0	0	2	0	-	R	E	A	D	
			-	P	R	I	N	T	.			
			R	E	A	D	-	-	-	-		
			M	O	V	E	-	-	-	-		
			W	R	I	T	E	-	-	-	-	
			0	0	2	0	-	E	X	I	T	
			E	X	I	T	.					

b. Verbs: Any reference to indentation implies four (4) columns.

CONT #	A	B
	8	12 16 20 24 28 32 36 40 44 48
		PERFORM 0030-READ-PRINT THRU 0030-EXIT.
	?	?
	?	?
	?	?
	?	?

```
0030-READ-PRINT.  
      READ IM-PERS-MASTER  
      AT END  
        MOVE 'ON' TO WS-IM-EOF-SWITCH.  
      MOVE ----  
      WRITE ---  
0030-EXIT.  
      EXIT.
```

[illegible]

CONT	A	B
7	8	12 16 20 24 28 32 36 40 44 48
		PERFORM 0090 -TABLE-BUILD THRU 0090 -EXIT
		VARYING WS-SUB FROM 1 BY 1
		UNTIL WS-SUB GREATER 36.

(2) IF.

(a) All IF statements will include the optional word THEN following the condition. The word THEN will be indented and appear on the next line. If the statement following the THEN is another IF (i.e., a nested IF), it will also be indented and appear on a separate line. The same rules apply to the use of the ELSE option.

(b) Examples: See Page 14.

(3) Go To. A GO TO statement may only be used to transfer control to a section exit or a paragraph exit.

(4) SORT. All clauses specified with the SORT verb will be indented and each clause will appear on a separate line. When either or both of the "INPUT PROCEDURE IS" and "OUTPUT PROCEDURE IS" options are used, each of the sections referred to must be clearly identifiable as a separate section. A work section will be used as the last section of the program, and it will contain individual routines (with appropriate exists) which are referenced thru the use of PERFORM statements in the INPUT PROCEDURE section or OUTPUT PROCEDURE section. Example: See Page 15.

(5) READ. AT END and INVALID KEY conditions associated with READ statements will be indented and appear on a separate line. Only one READ statement will be used for each input file.

IF Statement EXAMPLES

844

CONT	A	B	12	16	20	24	28	32	36	40	44	48	52	56	60	
			IF IM01-YEAR-OF-BIRTH LESS THAN 48													
			THEN PERFORM 0080-PT-TEST-ROSTER THRU 0080-EXIT.													
			}	}	}	}	}	}								
			IF IM01-YEAR-OF-BIRTH LESS THAN 48													
			THEN													
			IF IM01-MARITAL-STATUS EQUAL 'N'													
			THEN PERFORM 0040-PRINT THRU 0040-EXIT													
			ELSE PERFORM 0120-ADD THRU 0120-EXIT													
			ELSE													
			IF IM01-PT-SCORE LESS THAN 300													
			THEN PERFORM 0110-PT-FAIL THRU 0110-EXIT													
			ELSE Go To 0000-EXIT.													
			* NOTE: CORRESPONDING "THENS" AND "ELSE'S" ARE ALIGNED.													

14

IA-01-02-05

SORT Procedure EXAMPLES

CONT	A	B
8	12	16 20 24 28 32 36 40 44 48
		SORT SW-SORT-WORK-FILE
		ON ASCENDING KEY SW01-PART-NUMBER
		INPUT PROCEDURE IS 0020-IN-TO-SORT
		OUTPUT PROCEDURE IS 0030-SORT-OUT.
		0020-IN-TO-SORT SECTION.
		OPEN INPUT IM-PARTS-MASTER.
		PERFORM 0110-READ-RELEASE THRU 0110-EXIT
		UNTIL IM-EOF-SW EQUAL 'ON'.
		CLOSE IM-PARTS-MASTER.
		0020-SECTION-EXIT.
		EXIT.
		0030-SORT-OUT SECTION.
		OPEN OUTPUT OR-LISTING
		PERFORM 0120-RETURN-PRINT THRU 0120-EXIT
		UNTIL WS-SW-EOF-SWITCH EQUAL 'ON'.
		CLOSE OR-LISTING.
		0030-SECTION-EXIT.
		EXIT.
		0040-WORK SECTION.
		0110-READ-RELEASE.
		READ IM-PARTS-MASTER
		AT END -----

		0010-EXIT.
		EXIT.
		0120-RETURN-PRINT.

		0120-EXIT.
		EXIT.

(6) WRITE. Only one WRITE statement per record type is permitted. When writing to the printer, both the FROM and AFTER ADVANCING options may be used. The AFTER ADVANCING clause will be indented and appear on a separate line.

(7) The following verbs will not be used:

MOVE Corresponding
ALTER
GO TO DEPENDING ON
ENTER
ON

(8) Miscellaneous. All switches will be set up as three characters alphanumeric fields. The words "OFF" and "ON" will be used to change the status of a switch, or as a literal to check for a specific value in a switch.

1. INTRODUCTION.

a. ANSI COBOL, with its IBM extensions, has many advanced processing capabilities; that is, it has features that are not really needed to write most programs but can make it easier to write the programs, or make the programs themselves more flexible. Among those items which might be considered advanced processing are the following:

(1) The segmentation feature, which allows programs to be divided into segments.

(2) The source program library, which allows parts of programs to be saved in a library for later use.

(3) The debugging feature, which allows for a diagnostic printout while the program is executing.

(4) The report writer feature, which provides an alternate method of describing printed output files.

(5) The sort feature, which permits rearranging records of a file into any order.

(6) Index-sequential file handling, which allows processing of an indexed-sequential direct-access file.

b. We do not have the time nor space to discuss all these features, nor do all of them merit discussion. We will cover the two that seem to be used most often: the sort feature and index-sequential file handling. The others we will leave for you to explore on your own.

2. OBJECTIVE. You will learn the COBOL statements necessary to create, read sequentially, and update randomly an index-sequential file. You will also learn the statements and methods needed to use the sort feature. (You should note that index-sequential files are an IBM addition to the standard COBOL language. Although many other manufacturers also have adopted index-sequential files, the system with which you are working may not have this feature.) The two IBM operating systems in general used by the Army are DOS (Disk Operating System) and OS (Operating System). Where one or the other operating system has unique requirements, they are identified.

3. THE COBOL SORT FEATURE. The COBOL sort feature enables you to rearrange a file used by your program into any order you want. You can do the sorting as part of your program, instead of having to use a separate utility. If you have a file in social security number order, for example, you can resort it into alphabetical order by name, or by address or into whatever order you would like. To use the sort, we must set up a sort-file in our program. The purpose of a sort-file is to gather up all the records to be sorted; to hold them while the sort takes place, and to parcel the records out once they have been rearranged. The sort-file is essentially a work area into which the records to be sorted are placed. One method of sorting consists of three separate phases; each is described below.

a. Phase 1 - Input. The records to be sorted are read into the computer's memory, and "strings" of records are built and written to the sort-file. A "string" consists of a series of records in the desired sequence. As soon as the sequence is broken, a "string" is complete, and a new one is started. For example, let us say that records 4, 7, 8, 10, and 13 have been read and are put into a "string". The next record read is 9. The sequence has been broken, and a completed "string" of records 4, 7, 8, 10, 13 is now on the sort-file. A new "string" with record 9 as the first record is now started.

b. Phase 2 - Sort/Merge. After all records have been read and placed in "strings", two "strings" are brought back into memory at a time, and merged into one longer "string". This continues until only two "strings" are left.

c. Phase 3 - Output. The last two "strings" are merged when the records are passed back to memory or to an output device - whichever the program calls for.

Note: The number of "strings" which are merged at one time depends on what is known as the "merge order". The "merge order" is a function of the amount of memory available, number and type of input/output devices and number of work files to be used.

d. When using the COBOL SORT feature, input to and output from the SORT can be handled in one of several ways. First, let us look at what a programmer can do in the input phase. Basically, he has two options. They are:

- a. Copy every field of every record in the input file to the sort-file.
- b. Copy only selected fields and/or records in the input file to the sort-file.

If option "a" is selected, the input file is copied in its entirety. This method of input is useful when a file is simply to be rearranged, and all records are required for output. If option "b" is selected, only those records which need to be sorted are passed to the sort file. Also, individual fields which are not needed on output can be dropped, thereby making the sort record smaller. Removal of unnecessary records and fields reduces the amount of data to be sorted, and allows for better use of available memory. The time required for the sort itself will decrease. This option is normally used when printing a report from a file and

- a. the report must be in a different order than the file,
- b. only certain records are to be included and
- c. not all fields on the original record are needed on the report.

6. Output may be one of two options:

- a. Copy the sorted file to a permanent file;
- b. Read the sorted file as input and process it, one record at a time, like any other sequential input file.

Note: Option "a" is used when a permanent copy of the sorted file is desired and no reports or special processing are required. Option "b" is used when reports and/or special processing are required.

7. DESCRIBING THE SORT-FILE (Sort Work Area). Like any other file, we must have a SELECT statement for the sort-file. However, instead of having a corresponding FD in the Data Division, we must use a Sort Description (SD). Let us look at the SELECT first.

a. SELECT for a sort-file under IBM Disk Operating System (DOS) COBOL. The file must be either a tape or a disk file. It is divided into sections, each section on a different tape or disk area. The more sections into which the file is broken, the faster the sort will execute. If you are using disk as your work unit, you may have from one to eight sections. If you are using tape, you may have from three to nine. You may not mix disk with tapes. The choice of how many work units to use will depend mostly on how many tapes or disk areas are available. The format of the SELECT clause may be found in the IBM ANS COBOL Language Reference Manual. An example of a SELECT for a sort-file that will use four work units on disk is SELECT SORTING, ASSIGN TO 4 SYS001-UT-2314-S-SORTWK1. Note that the ASSIGN clause for the sort-file must specify SYS001. The unit may be either 2400 (tape) or 2314 or 2311 (disk). The -SORTWK1 parameter must be included when you are using standard labels. An example of a SELECT statement for a sort-file under OS is:

```
SELECT SORT-WORK-FILE ASSIGN TO 4 UT-S-SORTWK1
```

b. The sort description. In a program that uses a sort, each ordinary file must have an FD and the sort-file must have an SD. The format for the SD can be found in the IBM ANS COBOL Language Reference Manual under Sort File Description Entry.

Note: The sort-file-name must match the one given in the SELECT for the sort file, and the BLOCK CONTAINS clause may NOT be coded or used with an SD.

8. SORTING THE FILE. The file is sorted by executing a SORT statement in the PROCEDURE DIVISION. Each time the SORT is executed, the entire file is sorted. Usually, the SORT will be executed once. The SORT statement provides information about all phases of the sorting operation. The format of the SORT statement is shown below. The sort-file itself is never opened or closed by the programmer. Opening and closing of the sort-file is done automatically by the SORT verb. For the format of the SORT statement, refer to the IBM ANS COBOL Language Reference Manual. Let us see how each part of the SORT statement applies to a particular phase of the SORT operation.

a. Phase 1--Input. The INPUT PROCEDURE or USING clause (not both) determines how the sort-file will get the records that it is to sort.

(1) If the entire input file is to be sorted, you can use the USING file-name-2 option. This will cause the file-name-2 file to be copied onto the sort-file. You must not open, close, or read the file-name-2 file; copying is done automatically. (NOTE: file-name-2 cannot be open while the sort is executing.) When you code the USING, you do not have access to the records as they are being copied onto the sort-file. The record description under the SD must be identical in form to the record description under file-name-2.

(2) If you wish to manipulate the records from the input file before they are placed on the sort-file, you must use the INPUT PROCEDURE clause. By coding the INPUT PROCEDURE clause, you will cause the input phase to perform the section-name you give in the INPUT PROCEDURE clause.

(3) A section is made up of one or more paragraphs that are logically related. Every section is begun with a section name (a paragraph name followed by a space and the word SECTION). A section ends on the statement before the next section name or with the last card in the program, whichever occurs first. It is a good idea to have an exit paragraph (called section-exit in this text) at the end of each section. This allows the structured construction of the procedure division to be maintained.

(4) When the sort begins execution, the input procedure section is executed. The input procedure section is actually a self-contained program that:

- (a) opens the file that has the records to be sorted;
- (b) reads those records;
- (c) processes them in whatever way is needed;
- (d) passes the records to the sort file by using a RELEASE statement (explained later);
- (e) closes the input file;
- (f) and branches to the section exit.

Note: The input procedure section has its own driver paragraph (ending with a GO TO to the section-exit), any routines needed for processing the input records, and a section exit at the end of all the routines.

b. Phase 2--Sort/Merge. The SORT...ON...KEY portion of the sort statement controls the action of this phase. Once Phase 1 is finished, all the records that are to be sorted will be arranged in strings in the sort-work area. The SORT...ON...KEY will determine how the strings are to be merged. Rules on sort keys are listed below. If the file was in SSN order and we wish to sort it alphabetically by name, then NAME is the key on which we will be sorting. There can be more than one sort key; for example, we might want to sort the records by STATE and then by CITY within each STATE. In this case the STATE field would be the major sort key, and the CITY field would be the minor sort key. When you code the ON...KEY clauses, you must code the major key first, followed by the minor keys. For each key, you have the choice of sorting it

in ASCENDING (lowest first, highest last) sequence or DESCENDING (highest first, lowest last) sequence. (An alphabetical sort would be ASCENDING - A is first, Z is last). Rules on keys on DOS and OS:

(1) The sort keys that you are sorting on must be subdivisions of the record of the sort-file; i.e., they must come under the 01 which belongs to the SD.

(2) The maximum number of keys is 12.

(3) The total length of the keys must not exceed 256 bytes.

(4) Keys cannot contain or be subordinate to an OCCURS clause.

(5) Keys cannot be located after an OCCURS clause.

(6) All keys must be defined in the first 4092 bytes of the sort record.

c. Phase 3--Output. Once Phase 2 is finished, the records are in a sort work area and can be retrieved in the order in which you want them. The means of retrieving them is specified by the GIVING or OUTPUT PROCEDURE clauses.

(1) The GIVING file-name-3 operates in a similar manner to the USING; i.e., the sort-file is copied to an output file. The file that receives the sorted records must not be opened by the programmer when the sort is executed. File-name-3 will be opened and closed automatically. As with the USING, you do not have access to the records as they are copied from the sort-file to file-name-3. The record description of file-name-3 must be identical in form to the record description in the sort-file.

(2) By using the OUTPUT PROCEDURE clause, you may retrieve the sorted records one at a time and process them. Like the input procedure, the section-name you give will be performed. Within the section you can retrieve records from the sort-file by using a RETURN statement (discussed later in this text). The sort file must not be opened or closed. As in the input procedure section, the output procedure section is responsible for opening and closing the output file(s), processing the records returned from the sort, and branching to its section exit when all processing is complete. It should begin with a driver routine, be followed by all the routines needed to process the sorted records and end with its section-exit paragraph. Once the performance of the OUTPUT PROCEDURE is finished, phase 3 of the sort operation is done; and control is passed to the sentence following the sort verb.

9. THE RELEASE STATEMENT. The purpose of an INPUT PROCEDURE is to let you process the records before they are passed on to the sort-file. Once you have finished processing an input record in your INPUT PROCEDURE, you must use the RELEASE statement to place the record on the sort-file. The format of the RELEASE can be found in the IBM ANS COBOL Language Reference Manual. The RELEASE is the equivalent of a WRITE statement for use on sort-files. Even

the FROM option works in the same way as it does for a WRITE; i.e., if the FROM is used, the data-name indicated in the FROM option is moved to the sort-record area and then the sort-record is RELEASED. The RELEASE statement writes the contents of the sort-record onto the sort-file. It may only be used within the bounds of the section named in the INPUT PROCEDURE clause.

10. THE RETURN STATEMENT. The purpose of an OUTPUT PROCEDURE is to let you process the records after they have been sorted. The RETURN statement may only be used within the bounds of the section named in the OUTPUT PROCEDURE clause. If the RELEASE is the equivalent of a WRITE for a sort file, then the RETURN is the equivalent of a READ for a sort-file. The format of the RETURN is found in the IBM ANS COBOL Language Reference Manual. Where the RELEASE called for the name of the sort records, the RETURN calls for the name of the sort file. The INTO option and AT END clause operate in the same way as for a READ. If the INTO option is used, the record description of the output file must be identical in form to the record description of the sort file. The AT END condition will occur when all the records of the sort file have been returned. The statements between the words AT END and the period that ends the sentence will not be executed until the AT END condition occurs.

11. SORT-WORK SECTION. A WORK section following the input and output procedure sections is used to contain all the routines used in the program. The input and output procedures then PERFORM routines contained in the SORT-WORK SECTION. The SORT-WORK SECTION is used not only to preserve a structured program design, but also to define the end of the input and output procedure sections.

12. A sample program which uses the SORT verb is at Inclosure 1.

PRACTICAL EXERCISE

Personnel Listing

1. The Personnel Office has requested a listing of all personnel in the battalion who have completed more than 35 months of service.
2. You are provided with a report format and card layout.
3. The report will be generated from the personnel master file. The file is stored in card-image form on a 2314 disk file.
4. You will keep a count of the total number of people who have more than 35 months of service. In addition, only 20 lines are to be printed (assuming double spacing) on a page. If more than 20 lines need to be printed start over with a new page.

FORM LAYOUT

1										2										3										4										5										6										7																			
1234567890										1234567890										1234567890										1234567890										1234567890										1234567890										1234567890																			
NAME																				RANK																				SSN																				MON SER																			
XXXXXXXXXXXXXXXXXX																				XXX																				XXXXXXXXXX																				XXX																			
XXXXXXXXXXXXXXXXXX																				XXX																				XXXXXXXXXX																				XXX																			
XXXXXXXXXXXXXXXXXX																				XXX																				XXXXXXXXXX																				XXX																			
TOTAL NUMBER OF PERSONNEL WITH OVER 35 MONTHS SERVICE																																																												XXX																			

PERSONNEL MASTER INPUT FILE

<u>Record Position</u>	<u>Field Name</u>
1 - 2	Battalion
3	Company
4 - 12	Social Security Number
13 - 30	Name
31 - 32	Pay Grade
33 - 35	Rank
36 - 41	ETS
42 - 47	Date of Entry
48 - 50	Months of Service
51 - 52	Number of Dependents
53 - 80	Not Used

IA-01-05-08

DUMP DEBUG WORKSHEET

1. Hexadecimal Memory Location of Interrupt _____
2. Load Point _____
3. Relative Interrupt Address _____
4. ALC Instruction Interrupted _____
5. Corresponding COBOL Statement Number _____
6. COBOL Statement _____
7. COBOL Source Data Names
 Receiving Operand _____
 Sending Operand _____
8. Internal Data Name for
 Receiving Operand _____
 Sending Operand _____
9. Data Format of
 Receiving Operand _____
 Sending Operand _____
10. Length in bytes reserved by
 Receiving Operand _____
 Sending Operand _____
11. Machine Instruction Interrupted _____
12. Operation Code of Machine Instruction _____
13. Machine Instruction Format _____
14. Length of the Machine Instruction in bytes _____
15. How many Operand Lengths in Machine Instr _____
16. What is/are the Length(s) _____

17. Base Register for First Operand _____
18. Base Address for First Operand _____
19. Displacement for First Operand _____
20. Absolute Address of First Operand _____
21. Contents of Storage Allocated to First Operand _____
22. Base Register for Second Operand _____
23. Base Address for Second Operand _____
24. Displacement for Second Operand _____
25. Absolute Address of Second Operand _____
26. Contents of Storage Allocated to Second Operand _____

IF TEMPORARY STORAGE IS REFERENCED IN THE ALC INSTRUCTION (TS= in LISTX)
THEN FILL IN ITEMS 27 THRU 30.

27. Relative Address of TS _____
28. Absolute Address of TS _____
29. Internal Data Name of Item in Left Half of TS _____
30. Internal Data Name of Item in Right Half of TS _____
31. Origin of Data and /or Reason for Incorrect Format:
 - a. Input Record _____
 - b. Failure to Initialize _____
 - c. Failure to Generate _____

32. Corrective Action:

121-013-1413-150-A

2

IA-01-01-21

USAFPP 171-19 6/79

DEBUG

USAIA

SOFTWARE DIVISION, CSD

PRACTICAL EXERCISE NUMBER 2

PROGRAM NAME: DUMP-PF

C9L L18

BASIS C32DMP

```

00001 000010 IDENTIFICATION DIVISION.
00002 000020 PROGRAM-ID. CUMP-PE.
00003 000030 AUTHOR. LT HUDGIN.
00004 000040 REMARKS. PE DESIGNED FOR USING ALC TO DEBUG A COBOL PROGRAM.
00005 000050 ENVIRONMENT DIVISION.
00006 000060 CONFIGURATION SECTION.
00007 000070 SPECIAL-NAMES.
00008 000080 C31 IS CHAN1.
00009 000090 INPLT-OUTPUT SECTION.
00010 000100 FILE-CONTROL.
00011 000110 SELECT PERSONNEL-MASTER-FILE
00012 000120 ASSIGN TO SYS006-UT-2314-S.
00013 000130 SELECT PRINTER
00014 000140 ASSIGN TO SYS005-UR-1403-S.
00015 000150 DATA DIVISION.
00016 000160 FILE SECTION.
00017 000170 FD PERSONNEL-MASTER-FILE
00018 000180 LABEL RECORDS STANDARD.
00019 000190 01 IM01-PERSONNEL-MASTER-RCO.
00020 000200 05 IM01-SOC-SEC-NUM PIC X(9).
00021 000210 05 IM01-NAME PIC X(19).
00022 000220 05 IM01-DEPARTMENT PIC X(2).
00023 000230 05 FILLER PIC X(5).
00024 000240 05 IM01-NUMBER-DEPENDENTS PIC X(2).
00025 000250 05 IM01-MONTHLY-PAY PIC 9(4)V9(2).
00026 000260 05 FILLER PIC X(38).
00027 000270 FD PRINTER
00028 000280 LABEL RECORDS OMITTED.
00029 000290 01 OR01-PRINTER-RCO PIC X(133).
00030 000300 WORKING-STORAGE SECTION.
00031 000310 77 WS-MASTER-FILE-ECF-SWITCH PIC X.
00032 000320 77 WS-TOTAL-FAMILY-SIZE-SUM PIC S9(3)
00033 000330 77 WS-TOTAL-NET-PAY-SUM USAGE IS COMP-3.
00034 000340 77 WS-DEPARTMENT-FOLD PIC S9(4)V9(2)
00035 000350 77 WS-FEDERAL-TAX-HOLD USAGE IS COMP-3.
00036 000360 77 WS-FICA-TAX-HOLD VALUE IS SPACES.
00037 000370 77 WS-NET-PAY-HOLD PIC S9(4)V9(2)
00038 000380 77 WS-FAMILY-SIZE-HOLD USAGE IS COMP-3.
00039 000390 77 WS-TAX-RATE-HOLD PIC S9(3)
00040 000400 77 WS-FICA-TAX-RATE USAGE IS COMP-3.
00041 000410 77 WS-O-DEPENDENTS-TAX-RATE PIC S9(4)V9(2)
00042 000420 77 WS-O-DEPENDENTS-TAX-RATE USAGE IS COMP-3.
00043 000430 77 WS-O-DEPENDENTS-TAX-RATE PIC S9(3)
00044 000440 77 WS-O-DEPENDENTS-TAX-RATE USAGE IS COMP-3.
00045 000450 77 WS-O-DEPENDENTS-TAX-RATE PIC S9(3)
00046 000460 77 WS-O-DEPENDENTS-TAX-RATE USAGE IS COMP-3.
00047 000470 77 WS-O-DEPENDENTS-TAX-RATE PIC S9(3)
00048 000480 77 WS-O-DEPENDENTS-TAX-RATE USAGE IS COMP-3.
00049 000490 77 WS-O-DEPENDENTS-TAX-RATE PIC S9(3)
00050 000500 77 WS-O-DEPENDENTS-TAX-RATE USAGE IS COMP-3.
00051 000510 77 WS-O-DEPENDENTS-TAX-RATE PIC S9(3)
00052 000520 77 WS-O-DEPENDENTS-TAX-RATE USAGE IS COMP-3.

```

00053	0C052C			VALUE IS +.2.
00054	0C0530	77	WS-1-4-DEPENDENTS-TAX-RATE	PIC SV9(3)
00055	0C0540			USAGE IS COMP-3
00056	0C0550			VALUE IS +.15.
00057	0C0560	77	WS-5-8-DEPENDENTS-TAX-RATE	PIC SV9(3)
00058	0C0570			USAGE IS COMP-3
00059	0C0580			VALUE IS +.12.
00060	0C0590	77	WS-9-12-DEPENDENTS-TAX-RATE	PIC SV9(3)
00061	0C0600			USAGE IS COMP-3
00062	0C0610			VALUE IS +.08.
00063	0C0620	77	WS-13-99-DEPENDENTS-TAX-RATE	PIC SV9(3)
00064	0C0630			USAGE IS COMP-3
00065	0C0640			VALUE IS +.05.
00066	0C0650	77	WS-SWITCH-CA	PIC X
00067	0C0660			VALUE IS '0'.
00068	0C0670	77	WS-SWITCH-OFF	PIC X
00069	0C0680			VALUE IS '1'.
00070	000690	01	WS-COLUMN-HEADING.	
00071	0C0700	05	FILLER	PIC X(14)
00072	000710			VALUE IS SPACES.
00073	000720	05	FILLER	PIC X(18)
00074	0C0730			VALUE IS 'SSN'.
00075	0C0740	05	FILLER	PIC X(15)
00076	0C0750			VALUE IS 'NAME'.
00077	000760	05	FILLER	PIC X(13)
00078	0C0770			VALUE IS 'MONTHLY PAY'.
00079	0C0780	05	FILLER	PIC X(11)
00080	000790			VALUE IS 'NO OF DEPS'.
00081	0C0800	05	FILLER	PIC X(10)
00082	0C0810			VALUE IS 'PAY AVE'.
00083	0C0820	05	FILLER	PIC X(11)
00084	0C0830			VALUE IS 'FED TAX'.
00085	0C0840	05	FILLER	PIC X(11)
00086	0C0850			VALUE IS 'FICA TAX'.
00087	0C0860	05	FILLER	PIC X(19)
00088	0C0870			VALUE IS 'NET-PAY'.
00089	0C0880	01	WS-DETAIL-LINE.	
00090	000890	05	FILLER	PIC X(11)
00091	0C0900			VALUE SPACES.
00092	0C0910	05	WS-SOC-SEC-NUM	PIC X(9).
00093	0C0920	05	FILLER	PIC X(5)
00094	0C0930			VALUE IS SPACES.
00095	000940	05	WS-NAME	PIC X(18).
00096	0C0950	05	FILLER	PIC X(6)
00097	0C0960			VALUE SPACES.
00098	0C0970	05	WS-MONTHLY-PAY	PIC \$\$\$9.99.
00099	000980	05	FILLER	PIC X(6)
00100	0C0990			VALUE SPACES.
00101	0C1000	05	WS-NUMBER-DEPENDENTS	PIC Z9.
00102	001010	05	FILLER	PIC X(6)
00103	001020			VALUE IS SPACES.
00104	001030	05	WS-PAY-AVERAGE	PIC \$\$\$9.99.
00105	001040	05	FILLER	PIC X(2)

00106	001050		VALUE IS SPACES.
00107	001060	C5 WS-FEDERAL-TAX	PIC 99999.99.
00108	001070	05 FILLER	PIC X(5)
00109	001080		VALUE IS SPACES.
00110	001090	C5 WS-TAX-RATE	PIC V29.
00111	001100	05 FILLER	PIC X(7)
00112	001110		VALUE IS 'X'.
00113	001120	C5 WS-FICA-TAX	PIC 99999.99.
00114	001130	C5 FILLER	PIC X(3)
00115	001140		VALUE IS SPACES.
00116	001150	C5 WS-NET-PAY	PIC 99999.99.
00117	001160	C5 FILLER	PIC X(11)
00118	001170		VALUE IS SPACES.
00119	001180	C1 WS-FOOTING-LINE.	
00120	001190	C5 FILLER	PIC X(35)
00121	001200		VALUE IS SPACES.
00122	001210	C5 FILLER	PIC X(31)
00123	001220		VALUE IS
00124	001230		'DEPARTMENT TOTAL FAMILY SIZE'.
00125	001240	C5 WS-TOTAL-FAMILY-SIZE	PIC Z29.
00126	001250	C5 FILLER	PIC X(21)
00127	001260		VALUE IS ' AND TOTAL NET PAY'.
00128	001270	C5 WS-TOTAL-NET-PAY	PIC 99999.99.
00129	001280	05 FILLER	PIC X(34)
00130	001290		VALUE IS SPACES.
00131	001300	PROCEDURE DIVISION.	
00132	001310	OPEN INPUT PERSONNEL-MASTER-FILE	
00133	001320	OUTPUT PRINTER.	
00134	001330	MOVE WS-SWITCH-CFF TO WS-MASTER-FILE-EOF-SWITCH.	
00135	001340	PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT	
00136	001350	UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL TO WS-SWITCH-ON.	
00137	001360	CLOSE PERSONNEL-MASTER-FILE	
00138	001370	PRINTER.	
00139	001380	STOP RUN.	
00140	001390	0010-READ-AND-PRINT.	
00141	001400	READ PERSONNEL-MASTER-FILE	
00142	001410	AT END	
00143	001420	PERFORM 0020-TOTALS-RTN THRU 0020-EXIT	
00144	001430	MOVE WS-SWITCH-CN TO WS-MASTER-FILE-EOF-SWITCH	
00145	001440	GO TO 0010-EXIT.	
00146	001450	IF WS-DEPARTMENT-HOLD EQUAL TO SPACES	
00147	001460	PERFORM 0030-HEADING-RTN THRU 0030-EXIT	
00148	001470	ELSE	
00149	001480	IF WS-DEPARTMENT-HOLD NOT EQUAL TO IM01-DEPARTMENT	
00150	001490	PERFORM 0020-TOTALS-RTN THRU 0020-EXIT	
00151	001500	PERFORM 0030-HEADING-RTN THRU 0030-EXIT.	
00152	001510	IF IM01-NUMBER-DEPENDENTS EQUAL TO ZEROS	
00153	001520	MOVE WS-0-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD	
00154	001530	ELSE	
00155	001540	IF IM01-NUMBER-DEPENDENTS LESS THAN 5	
00156	001550	MOVE WS-1-4-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD	
00157	001560	ELSE	
00158	001570	IF IM01-NUMBER-DEPENDENTS LESS THAN 9	

```

00159 001580      MOVE WS-5-8-DEPENDENTS TAX-RATE TO WS-TAX-RATE-HOLD
00160 001590      ELSE
00161 001600      IF IM01-NUMBER-DEPENDENTS LESS THAN 13
00162 001610      MOVE WS-9-12-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00163 001620      ELSE
00164 001630      MOVE WS-13-99-DEPENDENTS-TAX-RATE TO
00165 001640      WS-TAX-RATE-HOLD.
00166 001650      MULTIPLY IM01-MONTHLY-PAY BY WS-TAX-RATE-HOLD
00167 001660      GIVING WS-FEDERAL-TAX-HOLD.
00168 001670      MULTIPLY IM01-MONTHLY-PAY BY WS-FICA-TAX-RATE
00169 001680      GIVING WS-FICA-TAX-HOLD.
00170 001690      SUBTRACT WS-FICA-TAX-HOLD WS-FEDERAL-TAX-HOLD FROM
00171 001700      IM01-MONTHLY-PAY
00172 001710      GIVING WS-NET-PAY-HOLD
00173 001720      DIVIDE WS-NET-PAY-HOLD BY WS-FAMILY-SIZE-HOLD
00174 001730      GIVING WS-PAY-AVERAGE.
00175 001740      MOVE IM01-SEC-SEC-NUM TO WS-SEC-SEC-NUM.
00176 001750      MOVE IM01-NAME TO WS-NAME.
00177 001760      MOVE IM01-MONTHLY-PAY TO WS-MONTHLY-PAY.
00178 001770      MOVE IM01-NUMBER-DEPENDENTS TO WS-NUMBER-DEPENDENTS.
00179 001780      MOVE WS-FEDERAL-TAX-HOLD TO WS-FEDERAL-TAX.
00180 001790      MOVE WS-TAX-RATE-HOLD TO WS-TAX-RATE.
00181 001800      MOVE WS-FICA-TAX-HOLD TO WS-FICA-TAX.
00182 001810      MOVE WS-NET-PAY-HOLD TO WS-NET-PAY.
00183 001820      ADD WS-NET-PAY-HOLD TO WS-TOTAL-NET-PAY-SUM.
00184 001830      ADD WS-FAMILY-SIZE-HOLD TO WS-TOTAL-FAMILY-SIZE-SUM.
00185 001840      WRITE C001-PRINTER-RCD FROM WS-DETAIL-LINE
00186 001850      AFTER ADVANCING 2 LINES.
00187 001860      0010-EXIT. EXIT.
00188 001870      0020-TOTALS-RTN.
00189 001880      MOVE WS-TOTAL-FAMILY-SIZE-SUM TO WS-TOTAL-FAMILY-SIZE.
00190 001890      MOVE WS-TOTAL-NET-PAY-SUM TO WS-TOTAL-NET-PAY.
00191 001900      WRITE C001-PRINTER-RCD FROM WS-FOOTING-LINE
00192 001910      AFTER ADVANCING 3 LINES.
00193 001920      MOVE ZEP03 TO WS-TOTAL-FAMILY-SIZE-SUM
00194 001930      WS-TOTAL-NET-PAY-SUM.
00195 001940      0020-EXIT. EXIT.
00196 001950      0030-HEADING-RTN.
00197 001960      MOVE IM01-DEPARTMENT TO WS-DEPARTMENT-HOLD.
00198 001970      WRITE C001-PRINTER-RCD FROM WS-COLUMN-HEADING
00199 001980      AFTER ADVANCING C001.
00200 001990      0030-EXIT. EXIT.

```


INTERNAL NAME	LVL	SOURCE NAME	BASE	DISPL	INTERNAL NAME	DEFINITION	USAGE
DNM=1-151	FD	PERSCNEL-MASTER-FILE	DTF=01		DNM=1-151		DTFSO
DNM=1-196	01	IM01-PERSONNEL-MASTER-RCO	BL=1	000	DNM=1-196	DS 0CL30	GROUP
DNM=1-234	02	IM01-SOC-SEC-NUM	BL=1	000	DNM=1-234	DS 9C	DISP
DNM=1-240	02	IM01-NAME	BL=1	009	DNM=1-260	DS 18C	DISP
DNM=1-279	02	IM01-DEPARTMENT	BL=1	019	DNM=1-279	DS 2C	DISP
DNM=1-304	02	FILLER	BL=1	010	DNM=1-304	DS 5C	DISP
DNM=1-320	02	IM01-NUMBER-DEPENDENTS	BL=1	022	DNM=1-320	DS 2C	DISP-NM
DNM=1-352	02	IM01-MONTHLY-PAY	BL=1	024	DNM=1-352	DS 6C	DISP-NM
DNM=1-378	02	FILLER	BL=1	02A	DNM=1-378	DS 38C	DISP
DNM=1-397	FD	PRINTER	DTF=02		DNM=1-397		DTFPR
DNM=1-428	01	CR01-PRINTER-RCO	BL=2	000	DNM=1-428	DS 133C	DISP
DNM=1-454	77	WS-MASTER-FILE-ECF-SWITCH	BL=3	000	DNM=1-454	DS 1C	DISP
DNM=2-000	77	WS-TOTAL-FAMILY-SIZE-SUM	BL=3	001	DNM=2-000	DS 2P	COMP-3
DNM=2-034	77	WS-TOTAL-NET-PAY-SUM	BL=3	003	DNM=2-034	DS 4P	COMP-3
DNM=2-067	77	WS-DEPARTMENT-HOLD	BL=3	007	DNM=2-067	DS 2C	DISP
DNM=2-095	77	WS-FEDERAL-TAX-HOLD	BL=3	009	DNM=2-095	DS 4P	COMP-3
DNM=2-124	77	WS-FICA-TAX-HOLD	BL=3	000	DNM=2-124	DS 4P	COMP-3
DNM=2-150	77	WS-NET-PAY-HOLD	BL=3	011	DNM=2-150	DS 4P	COMP-3
DNM=2-178	77	WS-FAMILY-SIZE-HOLD	BL=3	015	DNM=2-178	DS 2P	COMP-3
DNM=2-207	77	WS-TAX-RATE-HOLD	BL=3	017	DNM=2-207	DS 2P	COMP-3
DNM=2-233	77	WS-FICA-TAX-RATE	BL=3	019	DNM=2-233	DS 2P	COMP-3
DNM=2-259	77	WS-0-DEPENDENTS-TAX-RATE	BL=3	018	DNM=2-259	DS 2P	COMP-3
DNM=2-293	77	WS-1-4-DEPENDENTS-TAX-RATE	BL=3	010	DNM=2-293	DS 2P	COMP-3
DNM=2-329	77	WS-5-8-DEPENDENTS-TAX-RATE	BL=3	01F	DNM=2-329	DS 2P	COMP-3
DNM=2-365	77	WS-9-12-DEPENDENTS-TAX-RATE	BL=3	021	DNM=2-365	DS 2P	COMP-3
DNM=2-402	77	WS-13-99-DEPENDENTS-TAX-RATE	BL=3	023	DNM=2-402	DS 2P	COMP-3
DNM=2-440	77	WS-SWITCH-ON	BL=3	025	DNM=2-440	DS 1C	DISP
DNM=2-462	77	WS-SWITCH-OFF	BL=3	026	DNM=2-462	DS 1C	DISP
DNM=3-000	01	WS-COLUMN-HEADING	BL=3	028	DNM=3-000	DS 0CL122	GROUP
DNM=3-030	02	FILLER	BL=3	02C	DNM=3-030	DS 14C	DISP
DNM=3-069	02	FILLER	BL=3	036	DNM=3-049	DS 18C	DISP
DNM=3-068	02	FILLER	BL=3	049	DNM=3-068	DS 15C	DISP
DNM=3-087	02	FILLER	BL=3	057	DNM=3-087	DS 13C	DISP
DNM=3-106	02	FILLER	BL=3	064	DNM=3-106	DS 11C	DISP
DNM=3-125	02	FILLER	BL=3	06F	DNM=3-125	DS 10C	DISP
DNM=3-144	02	FILLER	BL=3	079	DNM=3-144	DS 11C	DISP
DNM=3-163	02	FILLER	BL=3	084	DNM=3-163	DS 11C	DISP
DNM=3-182	02	FILLER	BL=3	08F	DNM=3-182	DS 19C	DISP
DNM=3-201	01	WS-DETAIL-LINE	BL=3	0A8	DNM=3-201	DS 0CL133	GROUP
DNM=3-231	02	FILLER	BL=3	0A8	DNM=3-231	DS 11C	DISP
DNM=3-250	02	WS-SOC-SEC-NUM	BL=3	0B3	DNM=3-250	DS 9C	DISP
DNM=3-274	02	FILLER	BL=3	0BC	DNM=3-274	DS 5C	DISP
DNM=3-293	02	WS-NAME	BL=3	0C1	DNM=3-293	DS 18C	DISP
DNM=3-310	02	FILLER	BL=3	0D3	DNM=3-310	DS 6C	DISP
DNM=3-329	02	WS-MONTHLY-PAY	BL=3	0D9	DNM=3-329	DS 8C	NM-EDIT
DNM=3-364	02	FILLER	BL=3	0E1	DNM=3-364	DS 6C	DISP
DNM=3-383	02	WS-NUMBER-DEPENDENTS	BL=3	0E7	DNM=3-383	DS 2C	NM-EDIT
DNM=3-420	02	FILLER	BL=3	0E9	DNM=3-420	DS 6C	DISP
DNM=3-439	02	WS-PAY-AVERAGE	BL=3	0EF	DNM=3-439	DS 8C	NM-EDIT
DNM=3-474	02	FILLER	BL=3	0F7	DNM=3-474	DS 2C	DISP

MEMORY MAP

TGT	03549
SAVE AREA	00548
SWITCH	00550
TALLY	00554
SOFT SAVE	00558
ENTRY-SAVE	0055C
SOFT CORE SIZE	005A0
NSTC-REELS	005A4
SOFT SET	005A6
WORKING CELLS	005A8
SOFT FILE SIZE	00608
SOFT MODE SIZE	0060C
PGT-VN TBL	006E0
TGT-VN TBL	006E4
SOFTAB ADDRESS	006E8
LENGTH OF VN TBL	006EC
LNTH OF SOFTAB	006EE
PGM ID	006FC
A(INIT1)	006FB
UPSI SWITCHES	006FC
OVERFLOW CELLS	00704
BL CELLS	00704
CTFACR CELLS	00710
TEMP STORAGE	00718
TEMP STORAGE-2	00728
TEMP STORAGE-3	00738
TEMP STORAGE-4	00738
BLL CELLS	00738
VLC CELLS	0073C
SRL CELLS	0073C
INDEX CELLS	0073C
SUPADR CELLS	0073C
CNCTL CELLS	0073C
PFMCTL CELLS	0073C
PFMSAV CELLS	0073C
VN CELLS	00750
SAVE AREA =2	0075C
XSASH CELLS	0075C
XSA CELLS	0075C
PADAM CELLS	0075C
RPTSAV AREA	0075C
CHECKPT CTR	0075C
TOPTR CELLS	0075C

ERIAL POOL (PEX)

0C (LIT+0)	0C5C9C01	3C4C205B	20202120	48202040	202120F3	23212000
08 (LIT+24)	0C0C0C03	0C70C304	5B58C2C4	07C5D540	5B58C2C3	D3D6E2C5

POST	CC768
OVERFLOW CELLS	CC768
VIRTUAL CELLS	CC769
PROCEDURE NAME CELLS	CC76C
GENERATED NAME CELLS	CC764
SUBJECT ADDRESS CELLS	CC7C4
VNI CELLS	CC7C4
LITERALS	CC7CC
DISPLAY LITERALS	CC8C0

REGISTER ASSIGNMENT

REG 6 RL =3
 REG 7 PL =1
 REG 8 PL =2

132

```

000800
000800 58 10 D 1C8
000804 48 10 C 096
000808 94 BF 1 CC0
00080C 41 10 C 088
000810 58 00 D 1C8
000814 18 40
000816 05 F0
000818 50 00 F CC8
00081C 45 00 F C0C
000820 C0C00000
000824 0A 32
000826 41 10 C 088
00082A 58 00 D 1CC
00082E 18 40
000830 C7 00
000832 05 F0
000834 50 00 F C08
000838 45 00 F 00C
00083C 00C00000
000840 0A 02
000842 50 20 D 1C0
000846 58 80 D 1C0
00084A D2 00 6 C00 6 076
000850 58 00 D 2C8
000854 50 00 D 1F4
000858 58 00 C 01C
00085C 50 00 D 2C8
000860
000860 58 20 C 020
000864 05 00 6 00C 5 025
00086A C7 82
00086C 58 10 C 004
  
```

START

```

EQU *
L 1,1C3(0,13) DTF=1
SM 1,086(0,12) LIT+30
NI 000(1),X'BF'
LA 1,08A(0,12) LIT+32
L 0,1C8(0,13) DTF=1
LR 4,0
BALR 15,0
ST 0,008(0,15)
BAL 0,00C(0,15)
DC X'00000000'
SVC 2
LA 1,083(0,12) LIT+32
L 0,1CC(0,13) DTF=2
LR 4,0
BCR 0,0
BALR 15,0
ST 0,006(0,15)
BAL 0,00C(0,15)
DC X'00000000'
SVC 2
ST 2,1C0(0,13) BL =2
L 8,1C0(0,13) BL =2
MVC 000(1,6),026(6) DNM=1-454
L 0,208(0,13) VN=01
ST 0,1F4(0,13) PSV=1
L 0,01C(0,12) GN=01
ST 0,208(0,13) VN=01
EQU *
L 2,020(0,12) GN=02
CLC 000(1,6),025(6) DNM=1-454
BCR 8,2 DNM=2-44
L 1,004(0,12) PN=01
  
```

GN=01

DNM=2-46

DNM=2-44

134
135

155	CCC9D4 07 F1	GA=011	BCR 15,1		
	CCC9D6		EQU *		
	JCC9D6 F2 71 D 1DC 7 322		PACK 1D0(3,13),022(2,7)	TS=01	DNM=1-3
	JCC9D6 F9 10 D 1C6 C 069		CP 1D6(2,13),069(1,12)	TS=07	LIT+1
	JCC9E2 58 FC C C4C		L 15,0-010,12)	GN=013	
	00C9E6 07 AF		BCR 10,15		
156	JCC9E6 F8 71 D 1CC 6 013		ZAP 1D0(3,13),010(2,6)	TS=01	DNM=2-2
	0009EE F1 76 D 100 D 100		MVO 1D0(3,13),1D0(7,13)	TS=01	TS=01
	JCC9F4 F8 11 6 017 C 106		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
159	CCC9FA 58 10 C C48		L 1,04(0,12)	GN=012	
	J009FE 07 F1		BCR 15,1		
159	JCC9D2	GA=013	EQU *		
	J0CA00 F2 71 D 1DC 7 322		PACK 1D0(3,13),022(2,7)	TS=01	DNM=1-3
	J0CA06 F9 10 D 1C6 C 069		CP 1D6(2,13),069(1,12)	TS=07	LIT+2
	J0CA0C 58 FC C C50		L 15,050(0,12)	GN=014	
	00CA10 07 AF		BCR 10,15		
159	00CA12 F8 71 D 1CC 6 01F		ZAP 1D0(3,13),01F(2,6)	TS=01	DNM=2-3
	J0CA18 F1 76 D 100 D 100		MVO 1D0(3,13),1D0(7,13)	TS=01	TS=01
	J0CA1E F8 11 6 017 D 106		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
161	00CA24 58 10 C C48		L 1,04(0,12)	GN=012	
	00CA28 C7 F1		BCR 15,1		
161	J00A2A	GA=014	EQU *		
	000A2A F2 71 D 1DC 7 322		PACK 1D0(3,13),022(2,7)	TS=01	DNM=1-3
	J00A30 F9 11 D 1D6 C 059		CP 1D6(2,13),068(2,12)	TS=07	LIT+3
	000A36 58 FC C 054		L 15,054(0,12)	GN=015	
	000A3A 07 AF		BCR 10,15		
162	000A3C F8 71 D 1DC 6 021		ZAP 1D0(3,13),021(2,6)	TS=01	DNM=2-3
	000A42 F1 76 C 100 D 100		MVO 1D0(3,13),1D0(7,13)	TS=01	TS=01
	000A48 F8 11 6 017 D 106		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
164	000A4E 58 10 C 048		L 1,04(0,12)	GN=012	
	J0CA52 C7 F1		BCR 15,1		
164	00CA54	GN=015	EQU *		
	000A54 F8 71 D 1DC 6 023		ZAP 1D0(3,13),023(2,6)	TS=01	DNM=2-4
	000A5A F1 76 D 100 D 100		MVO 1D0(3,13),1D0(7,13)	TS=01	TS=01
	00CA60 F8 11 6 017 D 106		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
166	00CA66	GN=012	EQU *		
	000A66 F2 75 D 1DC 7 024		PACK 1D0(3,13),024(6,7)	TS=01	DNM=1-3
	000A6C FC 51 C 1D2 6 017		MP 1D2(6,13),017(2,6)	TS=03	DNM=2-2
	000A72 F1 76 D 1D0 D 100		MVO 1D0(3,13),1D0(7,13)	TS=01	TS=01
	000A78 F1 76 D 1D0 D 100		MVO 1D0(3,13),1D0(7,13)	TS=01	TS=01
	J0CA7E F8 33 6 C09 D 1D4		ZAP 009(4,6),1D4(4,13)	DNM=2-95	TS=05
	J00A84 94 0F 6 C09		NI 009(6),X'0F'	DNM=2-95	
168	000A88 F2 75 D 1D0 7 024		PACK 1D0(3,13),024(6,7)	TS=01	DNM=1-3
	000A8E FC 51 D 1D2 6 019		MP 1D2(6,13),019(2,6)	TS=03	DNM=2-2
	J00A94 F1 75 D 1C0 D 100		MVO 1D0(3,13),1D0(6,13)	TS=01	TS=01
	000A9A F8 33 6 00D D 1D4		ZAP 00D(4,6),1D4(4,13)	DNM=2-124	TS=05
170	000AA0 F8 73 D 1DC 6 00D		ZAP 1D0(3,13),00D(4,6)	TS=01	DNM=2-1
	J00AA6 FA 33 D 1D4 6 009		AP 1D4(4,13),009(4,6)	TS=05	DNM=2-9
	J00AAC F2 75 D 1D8 7 024		PACK 1D8(3,13),024(6,7)	TS=09	DNM=1-3
	000AB2 F8 33 C 1CC D 1D4		SP 1DC(4,13),1D4(4,13)	TS=013	TS=05
	000ABE F8 33 6 011 D 1DC		ZAP 011(4,6),1DC(4,13)	DNM=2-150	TS=013
	000ABE 94 0F 6 011		NI 011(5),X'0F'	DNM=2-150	
172	CC0AC2 F8 73 D 1D8 6 011		ZAP 1C8(3,13),011(4,6)	TS=09	DNM=2-1

CROSS-REFERENCE DICTIONARY

DATA NAMES	DEFN	REFERENCE
PERSONNEL-MASTER-FILE	00012	00132 00132 00137 00141 00141
INC1-SOC-SEC-NU	00021	00175
INC1-NAME	00022	00176
INC1-DEPARTMENT	00023	00145 00197
INC1-NUMBER-DEPENDENTS	00025	00152 00155 00158 00161 00178
INC1-MONTHLY-PAY	00026	00166 00168 00170 00177
PRINTER	00014	00132 00132 00137 00145 00191 00198
ORG1-PRINTER-RCO	00030	00185 00185 00195 00191 00191 00191 00191 00198 00198 00198 00198 00198
WS-MASTER-FILE-ECF-SWITCH	00022	00158 00198
WS-TOTAL-FAMILY-SIZE-SUM	00023	00134 00135 00144
WS-TOTAL-NET-PAY-SUM	00035	00184 00189 00193
WS-DEPARTMENT-HOLD	00037	00183 00190 00193
WS-FEDERAL-TAX-HOLD	00039	00146 00146 00146 00149 00197
WS-FICA-TAX-HOLD	00041	00166 00166 00170 00179
WS-NET-PAY-HOLD	00043	00168 00170 00181
WS-FAMILY-SIZE-HOLD	00045	00170 00170 00173 00192 00183
WS-TAX-RATE-HOLD	00047	00173 00184
WS-FICA-TAX-RATE	00049	00153 00156 00159 00162 00164 00166 00180
WS-0-DEPENDENTS-TAX-RATE	00051	00169
WS-1-4-DEPENDENTS-TAX-RATE	00054	00153
WS-5-9-DEPENDENTS-TAX-RATE	00057	00156
WS-9-12-DEPENDENTS-TAX-RATE	00060	00162
WS-13-99-DEPENDENTS-TAX-RATE	00063	00164
WS-SWITCH-ON	00066	00135 00144
WS-SWITCH-OFF	00068	00134
WS-COLUMN-HEADING	00070	00158 00198
WS-FOOTING-LINE	00089	00185 00185
WS-SEC-NU	00092	00175
WS-NAME	00095	00176
WS-MONTHLY-PAY	00098	00177
WS-NUMBER-DEPENDENTS	00101	00178
WS-PAY-AVEPAGE	00104	00173
WS-FEDERAL-TAX	00107	00179
WS-TAX-RATE	00110	00180
WS-FICA-TAX	00113	00181
WS-NET-PAY	00116	00182
WS-FOOTING-LINE	00119	00191 00191
WS-TOTAL-FAMILY-SIZE	00125	00189
WS-TOTAL-NET-PAY	00128	00190
PROCEDURE NAMES	DEFN	REFERENCE
0010-READ-ANC-PRINT	00140	00135
0010-EXIT	00187	00135 00145
0020-TOTALS-RTN	00189	00143 00150
0020-EXIT	00195	00143 00150

28/08/70	PHASE	XFR-AD	LCCRE	MICRE	LSK-AC	ESD TYPE	LABEL	LOADED	REL-FR
PHASE***	005000	005000	006213	2E 12 4	CSECT	DUMPOPE		005000	005000
					CSECT	IJGFIE4Z		005E28	005E28
					* ENTRY	IJGFIZ4Z		005E28	
					* ENTRY	IJGFIZZZ		005E28	
					* ENTRY	IJGFIEZZ		005E28	
					CSECT	ILBDSAEO		006000	006000
					ENTRY	ILBDSAEL		0060F0	
					CSECT	IJDFAPIZ		005010	005010
					* ENTRY	IJDFAZIZ		005010	
					CSECT	ILBDMYSO		0060C8	0060C8
					WXTN	STXITPSM			
					WXTN	ILBDDPG2			

* UNREFERENCED SYMBOLS

002 UNRESOLVED ADDRESS CONSTANTS

SSN	NAME	*MONTHLY PAY	NO OF DEPS	PAY AVE	FED TAX	FICA TAX	NET-PAY
-----	------	--------------	------------	---------	---------	----------	---------

05031 PROGRAM CHECK INTERRUPTION - FEX LOCATION 005ABE - CONDITION CODE 0 - DATA EXCEPTION
 OSC01 JOB DUMPPE1 CANCELED

GR 0-Y 00035082 00005A66 00035403 00005300 00035298 0000500E 000350E8 00005308
GR 8-F 00005400 0000509E 00005000 00005000 00005768 00005548 00035074 00005A54
COMREG 8C ACER IS 000368

[illegible][illegible]


```

Z0004E60 006337C9 35033991 39923A5C
C0003648 00000014 19201996 1A3C1A4C
CCCC0G0C 360400C0 00000000 03680089
CGG01A74 0C00000C 00C00C00 0C000000
CC000000 00000000 000005A0 00004030
CCCC01A74 0C0000CC 00C00CC0 0C000000
CC000000 00000000 000005A0 00004129

```

C7C9F000 E01E000F C4C2FC13 0130130F
C402F313 3133133F C402F413 4134134F
C402F714 2142142F E307F018 0180180F
E3C7F31E 31E3183F E307F418 4184184F
C505C44C 40000000 C505C44C 40004500

```

CCC00001 00000003 12000478 5858C2E3
5858C2E3 F1F0F5F0 00000001 00000003
CCC00003 13030303 5858C2E3 F2F2F6F0
F2F7F4F0 00000001 00000004 00010499
0002047F 5858C2D3 C9E2E3E5 C0000001
CCC00001 00000006 06303035 5858C2C4
5858C2C4 E40407C4 C0000001 00000006
CCC00006 090130CE 5858C2C4 C9E3C1D5
C9E3C6D7 00000001 00000006 080000E6
C904011A 5858C7E3 D7C9D240 C0000001
C0000001 00000006 08340042 5858C2C4

```

```

.....* .....!.....*
.....280878240 .....
*.....
.....
.....
.....
.....
.....
.....
.....

```

```

....
CRO....CPO.... PRO....DKO....
DK1....DK2.... DK3....DK4....
DK5.. ..DK6.... DK7....TP0....
TP1....TP2.... TP3....TP4....
TP5....TDO.... END ...END ...

```

```

.....
$BBDUMPB$BTMEBG .....$BT
1030.....1 $BT1050.....
...$BT1060.... .....$BT2260
.....$BT
$BT2848..... 2740.....
...$BLISTV....
.....$BDUMP .....$BD
UMPB.....Q $BDUMPD.....
...$BCLDAL... .....$BDITAN
.....F$BD ITFP.....M
$BXSCH..... .....$GTPIK ....
.....$BCRZAP .....$BD
UMPF.....*
```

--BG--

0C4FA0				0D0640D5	C104C540
004FC0	00005000	C0005768	00005548	80005C74	
004FEC	CC0C50CC	C0C052B8	5C005CCE	0C0050E8	
005000	05FC070C	5CCCF0CA	47FCFC8C	0C1050C0	
005020	FFFFFF7C	0CC05000	0C004F88	0C006870	
00504C	00C06FA8	CCC00368	D2C2733E	7299F912	
005060	73467318	47807014	D2117168	7276D202	
0C5080	C28372BA	58C0F0C6	58E0C000	58D0F0CA	
0050A0	47F0F0AC	58CEFO3A	90ECC00C	185D89F	
0050CC	000050CC	CC0C5000	00005768	C0005548	
0050E0	C4E404D7	F0C7C540	F1D6C902	C9D5C7F0	
005100	8CD9C520	CC150C12	0CC80C05	0CF0F1F0	
005120	C5404040	4C40404C	40404040	4C404040	
005140	D6D5E3C8	C3E840D7	C1E84040	C5D64CC6	
005160	40C6C5C4	4CE3C1E7	40404040	C6C9C3C1	
005180	40404040	40404040	4040E3C5	404C40C6	
0051A0	E8404C40	4C40404C	40404040	40407C1	
0051CC	40404040	--SAVE--			
0051EC	40404040	40404040	40404040	40404040	
005200	404C4040	4C404040	40404040	40404C40	
005220	404C4040	4C404040	40404040	40404040	
005240	E3D4C5D5	E340E3D6	E3C1C340	C6C1C4C9	
005260	C44CE3D6	E3C1D340	C5C5E340	C7C1E840	
005280	404C4C40	--SAVE--			
0052A0	000C0C50	CC0C0000	CC0C0000	CCCC0C00	

FF150007	C0005A94	00305C9E	00005000
CC005A54	00005C82	00005A66	000054C0
C00053D8	000054C0	00000000	017582AE
CCCC5C0C	000107FF	00305000	0001C784
0A0407F1	00005010	00305010	00005FA8
7341733E	474070C4	FA107348	73CFF911
7187728A	02087192	7260D202	71A37299
95CCE030	4770F0A2	9610D048	92FFE000
F08A9110	00480719	C7FF0700	00005C9E
CC005800	00005C84	C3D6C2C6	F0F0F0F1
F1000454	4CC7C540	C2C5C7C9	D5E24000
40404040	40404040	40404040	4040E2E2
D5C1D4C5	40404040	40404040	404040D4
C640C4C5	D7E240D7	C1E840C1	E5C540D4
40E3C1E7	404040D5	C5E360D7	C1E840D4
4C404040	4040404C	404040C5	E340D7C1
40404040	40404040	40404040	40404040

406C4040 404040D9 40404040 40404040
40404040 404040C8 40404040 40404040
40404040 4040404C 404040C4 C5D7C1D9
D3E840E2 C9E9C540 4040C4C5 D74CC1D5
4C40D3E8 40E2C9E9 C5404040 40404040

CCCCCCCC 00000000 00003234 0C000106

```

NO NAME
.....*.....
.....*.....
.....E*.....EY
..O.....O.....O.....E
.....E.....
.....K.....9
.....K.....K
K.....OF.....O
..O.....O.....
..E.....E.....
DUMPOPE 10RKINGO
RE.....O10
N
MONTHLY PAY NO O
FED TAX FICA
TE F
Y PA
.....*.....
.....*.....
.....Q.....
.....P.....P
.....E.....E
.....9
.....K.....K
.....O.....O
.....*
.....COBFOOO
1.....GE BEGINS
S
NAME
F DEPS PAY AVE
TAX NET-PAY
ET P

```

• R
 DEPA
 DEPA
 TMENT TOTAL FAMI LY SIZE
 D TOTAL NET PAY LY SIZE

.....

CUMPF1

2F/CR/78

0052C0	000C5320	3CC05340	040C5528	2C0EE2E9	E2F0FCF6	40010300	00000000	00028020SY	S006
0052EC	000C0000	06005308	80C00000	000C0946	000C0000	00460000	02005904	00080013Q.....H.....
0053C0	2890CC4F	CC46CC0C	010C60F0	5821CC58	000C5308	0300005C	00005427	8E0060D0-0.....Q.....-.....
005320	C70052F2	4C000006	310052F4	4000CC05	C900532F	03000000	06005390	00000050	...2.....4...
005340	2800CC4F	CC000000	000C0000	00000105	000C537C	03005378	00005010	088488F11.....
005360	010054C1	CC000000	070C4120	00000000	C10C5431	20000084	06005418	00000050	...A.....f.....
005380	F6F3F9F1	F6F7F0F3	F9C8C6E3	E3400900	C2E84040	40404040	40404040	F1F0F1D7	639167039HOTT RO	8Y	01018
0053A0	C6C3FCF9	F0F5F3	F0FC4040	40404040	4C404040	40404040	40404040	40404040	FC090398000.....	78476402
0053C0	4C404040	40404040	40404040	40404040	CC1C58E0	F064C7FE	F7F3F4F7	F6F4F0F2	5GRUNDY GARVEY J	9	01015FC120568
0053E0	F5C7D9E4	F5C4F40	C7C1C9E5	C5E840D1	C54C40F0	F1F0F1E2	C6C3F1F2	F0F5F6F8	000.....	1
00540C	F0F0C4C0	4C404040	4C404040	4C404040	4C404040	40404040	40404040	40404040	N0.....	SS
00542C	40404040	4C404040	E0004770	FC5A47F0	F14C4040	40404040	40404040	4040E2E2	MONTHLY PAY NO O	NAME	M
005440	C5404040	4C404040	4C404040	4C404040	C5C1C4C5	40404040	40404040	40404040	FED TAX FICA	F DEPS PAY AVE	
005460	C4C5E3C8	C3E94C07	C1E94040	D5D64C06	C64C4C5	D7E240D7	C1E340C1	E5C54040	TAX NET-PAY00.....	
00548C	40C6C5C4	4C53C1E7	4C4C4040	C6C9C3C1	4C4C4C4C	40000000	47F3FC4C	0A320000	00.....	
0054A0	40404040	4C404040	40404040	40404040	F3F991E0	10024710	F0240A37	90CEFO6000.IJDFCZZW	39.....0.....0-	
0054C0	0A320000	47F0F01A	C9D1C4C6	C3E9E9E6	98CEFO60	02001028	1017D200	101710160.....	..0-K.....K.....	
0054E0	58E01C1E	48C0102E	06C018D0	44C0F06C	1C28C40C	C7FE0000	00000000	000000000.....	
0055C0	CA0CC7FE	F19D1C02	471CFC56	0A074200	C9D1C3C6	E9C9E9F0	030A0A00	91801002K.....0.....	IJCFZIZO.....	
00552C	00000000	C7C05000	0000F01A	0A320000	1C044780	F24C914C	10024710	F04658E0	..0.....0.....	
005540	471CFC2E	0A0750E0	00C6459E0	10209101	ECC04770	F05A47F0	F04658E0	F06407FE	0..00.....N..0-0..00.....0..	
005560	F06447F0	F01A5FE0	101C07FE	C501FC60	3C0C004F	00000000	00000000	00005800	/.....00.....	
005580	E15C0000	CC000000	00C0F04C	0A32C000	FE001118	4PC01C2E	06C018D0	44C0F06C0.....0-0.....	
0055A0	00000000	CC000000	FC240AC7	90CEFC60	C000004F	00000000	00000000	00000000	..0-K.....K.....	
0055CC	98CEFO6C	D2001029	1C17C200	10171016	CCCC0000	00000000	00000000	00000000	
0055EC	CC000000	CC000000	CC000000	CC000000	CCCC0000	00000000	00000000	00000000	
0055G0	CCCC0000	--SAME--			CCCC0000	00000000	00000000	00000000	
0055H0	CCCC0000	CCCC0000	CCCC0000	CCCC0000	CCCC0000	00000000	00000000	00000000Q.....	
0055I0	00000000	00005308	000054C0	000050E8	CCCC0000	00000000	00000000	00005800Y.....	
0055J0	CCCC0000	CCCC0000	CCCC0000	CCCC0000	CCCC0000	00000000	00000000	00005800Y.....	
0055K0	00005C00	CCCC05C8	CCCC0C470	CCCC000E8	00005806	0000580C	00005C4E	00005C7C	..E...*.....Y*+.....Y	
0055L0	C00003C8	00000398	000060C8	000058D4	CCCC0592E	0000591A	00005964	00005956-H...M	..80..*.....*	
0055M0	C0005C82	CCCC05860	00005872	CCCC05904	00005A66	00005A00	00005A2A	00005A54	..*.....-.....	
0055N0	000059AC	C0005996	CCCC05944	000059D6	0F5C9C01	3C402058	20202120	482020400.....	
0055O0	C0005C3C	C00058DC	CCCC05C4E	00005C82	5858C2D6	D7C50540	5858C2C3	D3D6E2C5	..0.....	..80OPEN ..80CLOSE	
0055P0	202120FC	20212000	CCCC0000	CCCC00004	58C0C1C8	184005F0	5C00F308	4500F00C	..JH.....	..JH..0E..0.....0..	
0055Q0	581CC1C8	4810CC86	948F100C	4110CC88	C7CC05F0	5000F30C	4500F30C	00005348J..	..0E..0.....0.....	
0055R0	00005298	CA024110	CCCC05800	D1CC1840	5800D208	5000D1F4	5800C01C	5000D208	..E..J...J..K..-..	..K..E..J4...E..K..	
0055S0	0A025020	D1C05880	D1C0C200	60006026	C7F15800	D1F45000	D2085800	D1C81840	..N..-.....	..1..J4E..K...JH..	
0055T0	5820C02C	C5006000	60250732	5810CC04	CCCC0000	0A025810	D1C84610	C08605F00E..0.....0..JH.....C	
0055U0	4110C090	C70005F0	5000F0C8	4500F00C	D2DF1000	20005800	D1CC1840	4110C0900.....	K.....J..	
0055V0	918010C0	4780F016	41101004	412010E0	GA020A0E	5810D1C8	58F0C024	91201010	..0E..0.....0.....JH..0.....	
0055W0	C70C05FC	5C00F008	4500F00C	00000000	101045E0	F0085020	D18C5870	D18C58F00..K...0..0..0..E..J...J...C	
0055X0	C71F1841	41F0C024	D2021041	F00158F0	50C0D20C	5810C00C	07F15800	D1F85000K..E..J8...	E..K.....1..J8E..	
0055Y0	C02807FF	58C0D20C	5C00C1F8	5800C02C	C030954C	60070772	D5006008	60070772	K..K..-.....1..	..-.....N..-.....	
0055Z0	D20C02C0	60006025	5810C008	07F15820	5810C010	07F15800	D1F85000	D2105810	..K..E..J.....E..K..1..J..E..K.....	
0055A1	5800D210	5C00D1FC	5800C034	5C00D210	*800D20C	5000D20C	5800C03C	5000D20C	..1.....N..-.....	..K..E..K.....E..K..	
0055B1	C03807F1	5820C038	D501C0C7	7018C782	J2105000	D2045800	C0405000	D21058101..K..E..K...	K..E..K.....E..K...	
0055C1	5810CC0C	C7F15800	D20C500C	D2CC5800	7022F910	D1D6C068	58F0C044	077FF871	..1..K..E..K..2..J..	..9..J0...0.....8..	
0055D1	C01007F1	5800D204	5000C210	F271D1D0	5810CC48	07F1F271	D1D87022	F910D1D6	J..-1..J..J..8..-..J012..J...9..J0	
0055E1	D1C06C18	F176D1D0	D1D0F811	6017C1D6	D1D0C010	F8116317	D1D85810	C04807F1	..0.....8..J..-1..	J..J..8..-..J0.....	
0055F1	C06558FC	CC4C07AF	F871C1D0	E01D1F76	G7AFF871	D1D0601F	F176D1D0	D1D0F811	2..J...9..J0...0..E	..8..J..-1..J..J..8..	
0055G1	F271D1D0	7022F910	D1D6C06A	58F0C050	F911D1C6	C06858F0	C05407AF	F871D1D0	-..J0.....12..J...	9..J0...0.....8..J..	
0055H1	6C17D1D6	5810C049	07F1F271	D1D07022							

285

005440	6321F176	C1000100	F8116017	D1065910	C04907F1	F8710100	6023F176	D1000100	-1.J.J.8.-J0..	...18.J.-1.J.J.
005460	F8116017	C106F275	D1007024	F0510102	6017F176	D1000100	F1750100	D100F833	8.-J02.J.....JK	-1.J.J.1.J.J.8.
005480	60090104	940F6009	F2750100	7024F051	C1026219	F1750100	D100F833	6000D104	-JM.-2.J.....	JK-1.J.J.8.-JM
0054A0	F8730100	6009F633	D1046009	F2750100	7024F833	D1000104	F8336011	D100940F	8.J.-...JM-2.JQ	...J.JM8.-J...
0054CC	6011F877	C1066011	F0510104	6015F673	C103010A	D2090100	C0604110	D1E60F09	-8.J0-...J.-8.	JQJ.K.J.....JW..
0054EC	C1E0010C	61092525	1000E207	60E0D1E2	12056033	70000211	60017009	F2750100	J.J.....K.-JS	K.-...K.-A..2.JC
0054FC	70240209	01000000	4110010A	0F090100	C1000100	92591000	D2076009	D1E2F271	..K.J.....JW..J.	J.....K.-RJS2.
0054FC	D1070222	C2030100	C0770503	D1E0010E	C2016007	D1E20209	D1000060	4110010E	JQ..K.J.....J.J.	K.-XJSK.J.....JW
0054FC	C0900100	60090100	92591000	C2076009	C1E2F871	D1046017	D2050100	D108940F	..J.-...K.-9	J58.JQ-P.JQJQ..
0054FC	D10E0203	D1E0007E	0E030100	C10E0201	61060102	D2090100	C0604110	D1E60F09	J.K.J.....J.J.K.	/JJK.J.....JW..
0054FC	C1E00000	61092525	10000207	610F0102	E2090100	C0604110	D1E50F09	D1E60011	J.-...K.-/JS	K.J.....JW..J.-
0054FC	06109258	10000207	61140102	FA336003	6011F611	60016015	02548000	60485810	..K.-/JS.-	-...-K.-...
0054FC	D100020C	60000500	10104500	F0000200	C1000500	D1000500	D2030701	D2030100	J..0...0...0..	J...J...K...K..J.
0054FC	C0770503	C1E00001	C2026172	D1E1F873	C1046003	D2030100	D103940F	D1000209	...J.-K.-/J.8.	JQ-P.JQJQ..J.K.
0054FC	C1E00000	4110010E	0F090100	C1000010	92591000	C207610A	D1E20203	80006130	J.....JW..J.J.	..K.-/JJK.../.
0054FC	92408004	50100100	92590000	58F01010	45000000	50200100	58000100	D2016001	..J.-...0..	..0..J...J.K.-
0054FC	C0770503	60000000	58100200	07F10201	60077010	D2790000	60239240	807AD209	..K.-...K...K.	-...K.-...K.
0054FC	0670007A	50100100	92F10000	58F01010	45000000	50200100	58000100	58100210	...J...l...0..	..0..J...J.K.
0054FC	C7F10A0E	50000000	50500004	58200000	95000000	077992FF	20000610	D04850E0	..l...l...l...	...l...l...l...
0054FC	005405FC	91200048	47E0F016	58000048	98200000	58000054	07F96200	D0484160	...0...0...	...l...l...l...
0054FC	00044110	C0044170	C0680070	05000040	10000100	50401000	87165000	41800100	...J...l...l...	...K...K...-JD..
0054FC	41700100	C0105800	80001000	50000000	07261000	D2090200	C0505860	D1C45870	J...J...l...0	...00...00..
0054FC	D1000500	C1005800	C05407FE	70000500	C0220000	47F0F004	0A320000	47F0F01A	IJDFAP1239...	0...0...0...K.
0054FC	C0010000	C1000000	F3F99100	10024710	F0240A07	90000000	58001018	06E00200	...0...0...	..07...00..0..
0054FC	1017E000	10000000	43001017	41E00010	43000000	19004780	F05046E0	F0400A32	..l...l...l...	...0...0...
0054FC	43000000	19004780	F0769180	F0769180	10164780	F0769200	10280A00	91801002	..0...l...0...	...0...0...
0054FC	47100000	0A074200	10284200	10160A00	91801002	4710F08A	0A070201	10261003	..0...l...0...	...0...0...
0054FC	92011028	58001000	D2021029	10195000	10180000	98000000	44001022	58000004	...K...l...0.	...0...0...04
0054FC	0A00007E	51801002	4710F08E	0A079101	10154730	F00494FE	10159102	1026078E	...0...0...	..0M...0...
0054FC	47F0F00A	51011027	C78E1200	4780F0E4	18F007FF	92801028	0A0007FE	00005768	..00...0...0U	..0...0...0...
0054FC	00005548	00000074	F2C201F9	F8F7F6F5	F4F00000	45000000	93080000	C38883A8	...28A98765	43C1+-0..L.C...
0054FC	A398E380	C3181373	47F0F014	47F0F252	47F0F022	47F0F014	47F0F014	0A320001	..T...00..02.	..00..00..00..1J
0054FC	C7C60000	50000000	C0080000	F2600180	10494710	F04A58A0	102000A0	10584AA0	GFIWZ39...2..	...0...0...0...
0054FC	104A5CA0	10600140	10274710	F10047F0	F0A09880	10588680	F08B5090	10589110	..l...-...l...0	0...0...0...0...
0054FC	10154780	F07E1800	41A00100	10CA47C0	F07602FF	00000000	1A0A868A	F0644100	...0...0...	0K...0...0...
0054FC	C0F04400	F25A98AE	F2604400	105407FE	91001049	4780F098	94F71049	47F0F100	...2...2...	...0...0...0...
0054FC	91081004	4710F0A2	0A000000	10024710	F0AC0A07	91011004	4780F088	47F0F236	...0...0...	0...0...0...02.
0054FC	91091003	4780F0C4	91A01004	4780F0C0	58001004	47F0F0E4	91921003	4780F110	...0J...0...	...00U...0...
0054FC	91801004	4780F14C	58001004	98AD026C	90F1F100	58101080	0700050E	98F1E000	...l...l...2.	..l...l...l...
0054FC	90AC026C	47F0F154	00000000	00000000	C0000000	00000000	91401005	4780F158	..2..01...0...	...0...0...0...
0054FC	48910000	4890F260	4780F13C	4880105E	4890F260	4720F124	4740F13C	96401049	...2...01...	..2...l...l...
0054FC	47F0F158	91041004	4780F0D8	58001050	47F0F0E4	91201004	4780F158	96101049	..01...0Q...0	..00U...l...0...
0054FC	58801000	41800000	50801058	91401049	4710F178	4A80104A	50801060	47F0F18A	...l...l...01.	..l...l...l...
0054FC	4800104A	45F00000	1AC85000	106009740	10454800	103489E0	000B4300	F2634180	...A...l...-	...2...2...
0054FC	00051800	40001000	43081043	19004770	F18642EB	103888E0	000B4690	F1904100	...N...2...	...2...2...2.
0054FC	00014200	10380503	10301036	4700F202	96081049	9001F280	98AD026C	4100F284	..2...2...0...	0...l...0...
0054FC	4110F264	CAC29801	F2809120	10104710	F07E9140	1027478C	F1FA948F	102747F0	0...2...0...	2...K...0...
0054FC	F0AC010B	10494710	F21A9108	10644780	F21A5880	10800202	10811020	50801020	...2...0...	0...0...0...
0054FC	0A009110	10494780	F22A9710	104947F0	F0285680	10499880	105847F0	F0569180	...l...l...l...	...01...K...-
0054FC	10264710	F1009120	10274710	F10094FF	10495640	102747F0	F10094FF	10581060	..K...6...	PEN...l...l...
0054FC	C7F0D200	C0000000	00000001	5858020A	C7C0D540	00005000	00005000	00005768	...0...0...	...00..1805AE031
0054FC	C0005548	000058F6	00000000	C0000000	F0C00000	00000000	00000000	00000000	...6...0...	...00..00..02..
0054FC	C0000000	C0000000	F0081049	4780F0C9	C5F047F0	F0190903	C204E201	C5F0F3F1	4303/07/74...00.	...00..00..02..
0054FC	F4F3FCF3	61F0F761	F7F41A00	47F0F022	41000001	05F05000	F0E00000	F0F24144		

006102	00005110	40144783	F030910E	4015471C	F030912C	40024780	F02C4130	001047F0
006120	F0335811	00044154	00004850	F11A5E55	CCCC4155	00001255	4783F0AC	1835D501
00614C	F0E43CCC	4780FC5A	4133C002	47F0F049	5C50F0EA	D5033006	F0E34730	F09E4403
00616C	00064403	CCCC1E52	92FC5000	D2CE50C1	5CCC123C	4780F05A	4903F0E6	4780F092
006180	56015002	47F0F09E	56015001	47F0F09E	56315030	59E0F0EE	9835F0F2	58F0F0EA
0061A0	07FF89C0	C0184840	F11C1604	4110F13A	C1C2910E	D049478C	F00C5810	F11258F0
0061C0	F1160701	1CC21C32	4111003C	50D10330	C7FF1900	0A060000	000335F0	00011039
0061E0	47C91C43	19CD4770	F18642E8	1C388E5C	CCC84680	F19C41CC	00014208	1038D503
006200	5E59C2C3	66C2CE59	3CC00300	00000000	00040038	00000000	00033030	00000000
00622C	00CCCC0C	--SAME--						
0107E0	C03C0C00	C000C000	0000C000	CCCCCCCC	CCCCCCCC	00000000	00C30300	00C04000

```

0... ..0... .. 0... ..0... ..0
0.....E1.. .. .....0...N.
OU...0.....00. EEO.N...0...0...
.....OE,K,E. E....0...OM...O.
..E..00...E..00. ..E...0...OZ.OO.
.....l...l... .. ....0...l..O
lp.....EJ.. .. ....0...
.....l..... .. ,l.....N.
$BOBER..... ..
.... ..
..... ..

```

D E B U G

USAIA

SOFTWARE DIVISION, CSD

PRACTICAL EXERCISE NUMBER 3

PROGRAM NAME: DUMP-PE

IA-02-02-22

1

288

```

CBL LIP
00001      BASIS C32DMP
00002      C0001C IDENTIFICATION DIVISION.
00003      C0002C PROGRAM-ID. DUMP-PF.
00004      C0003C AUTHOR. LT HUDGIN.
00005      C0004C REMARKS. PE DESIGNED FOR USING ALC TO DEBUG A COBOL PROGRAM.
00006      C0005C ENVIRONMENT DIVISION.
00007      C0006C CONFIGURATION SECTION.
00008      C0007C SPECIAL-NAMES.
00009      C0008C      CO1 IS CHAN1.
00010      C0009C INPUT-OUTPUT SECTION.
00011      C0010C FILE-CONTROL.
00012      C0011C      SELECT PERSONNEL-MASTER-FILE
00013      C0012C      ASSIGN TO SYS006-UT-2314-S.
00014      C0013C      SELECT PRINTER
00015      C0014C      ASSIGN TO SYS005-UR-1403-S.
00016      C0015C DATA DIVISION.
00017      C0016C FILE SECTION.
00018      C0017C FD PERSONNEL-MASTER-FILE
00019      C0018C LABEL RECORDS STANDARD.
00020      C0019C 01 IM01-PERSONNEL-MASTER-RCO.
00021      C0020C      C5 IM01-SOC-SEC-NUM          PIC X(9).
00022      C0021C      C5 IM01-NAME                PIC X(18).
00023      C0022C      C5 IM01-DEPARTMENT          PIC X(2).
00024      C0023C      C5 FILLER                    PIC X(5).
00025      C0024C      C5 IM01-NUMBER-DEPENDENTS   PIC 9(2).
00026      C0025C      C5 IM01-MONTHLY-PAY         PIC 9(4)V9(2).
00027      C0026C      C5 FILLER                    PIC X(38).
00028      C0027C FD PRINTER
00029      C0028C LABEL RECORDS OMITTED.
00030      C0029C 01 OR01-PRINTER-RCO              PIC X(133).
00031      C0030C WORKING-STORAGE SECTION.
00032      C0031C 77 WS-MASTER-FILE-EQF-SWITCH   PIC X.
00033      C0032C 77 WS-TOTAL-FAMILY-SIZE-SUM     PIC S9(3)
00034      C0033C                                     LSAGE IS COMP-3.
00035      C0034C 77 WS-TOTAL-NET-PAY-SUM          PIC S9(5)V9(2)
00036      C0035C                                     USAGE IS COMP-3.
00037      C0036C 77 WS-DEPARTMENT-HOLD            PIC X(2)
00038      C0037C                                     VALUE IS SPACES.
00039      C0038C 77 WS-FEDERAL-TAX-HOLD          PIC S9(4)V9(2)
00040      C0039C                                     USAGE IS COMP-3.
00041      C0040C 77 WS-FICA-TAX-HOLD              PIC S9(4)V9(2)
00042      C0041C                                     LSAGE IS COMP-3.
00043      C0042C 77 WS-NET-PAY-HOLD               PIC S9(4)V9(2)
00044      C0043C                                     LSAGE IS COMP-3.
00045      C0044C 77 WS-FAMILY-SIZE-HOLD          PIC S9(3)
00046      C0045C                                     LSAGE IS COMP-3.
00047      C0046C 77 WS-TAX-RATE-HOLD             PIC S9V9(2)
00048      C0047C                                     LSAGE IS COMP-3.
00049      C0048C 77 WS-FICA-TAX-RATE            PIC S9(3)
00050      C0049C                                     VALUE IS .058
00051      C0050C 77 WS-O-DEPENDENTS-TAX-RATE    PIC S9(3)
00052      C0051C                                     LSAGE IS COMP-3.
00053      C0052C

```


00053	000510		USAGE IS COMP-3
00054	CCC52C		VALUE IS +.2.
00055	000530	77 WS-1-4-DEPENDENTS-TAX-RATE	PIC SV9(3)
00056	000540		USAGE IS COMP-3
00057	CCC55C		VALUE IS +.15.
00058	000560	77 WS-5-8-DEPENDENTS-TAX-RATE	PIC SV9(3)
00059	00057C		USAGE IS COMP-3
00060	00058C		VALUE IS +.12.
00061	00059C	77 WS-9-12-DEPENDENTS-TAX-RATE	PIC SV9(3)
00062	00060C		USAGE IS COMP-3
00063	CCC61C		VALUE IS +.08.
00064	000620	77 WS-13-59-DEPENDENTS-TAX-RATE	PIC SV9(3)
00065	000630		USAGE IS COMP-3
00066	00064C		VALUE IS +.05.
00067	000650	77 WS-SWITCH-CN	PIC X
00068	00066C		VALUE IS '0'.
00069	000670	77 WS-SWITCH-OFF	PIC X
00070	00068C		VALUE IS '1'.
00071	00069C	01 WS-COLUMN-HEADING.	
00072	00070C	05 FILLER	PIC X(14)
00073	00071C		VALUE IS SPACES.
00074	000720	05 FILLER	PIC X(18)
00075	00073C		VALUE IS 'SSN'.
00076	00074C	05 FILLER	PIC X(15)
00077	000750		VALUE IS 'NAME'.
00078	00076C	05 FILLER	PIC X(13)
00079	000770		VALUE IS 'MONTHLY PAY'.
00080	00078C	05 FILLER	PIC X(11)
00081	000790		VALUE IS 'NO OF DEPS'.
00082	000800	05 FILLER	PIC X(10)
00083	00081C		VALUE IS 'PAY AVE'.
00084	000820	05 FILLER	PIC X(11)
00085	00083C		VALUE IS 'FED TAX'.
00086	000840	05 FILLER	PIC X(11)
00087	000850		VALUE IS 'FICA TAX'.
00088	00086C	05 FILLER	PIC X(19)
00089	000870		VALUE IS 'NET-PAY'.
00090	000880	01 WS-DETAIL-LINE.	
00091	000890	05 FILLER	PIC X(11)
00092	000900		VALUE SPACES.
00093	000910	05 WS-SCC-SEC-NUM	PIC X(9).
00094	000920	05 FILLER	PIC X(5)
00095	000930		VALUE IS SPACES.
00096	000940	05 WS-NAME	PIC X(18).
00097	00095C	05 FILLER	PIC X(6)
00098	000960		VALUE SPACES.
00099	000970	05 WS-MONTHLY-PAY	PIC \$\$\$9.99.
00100	00098C	05 FILLER	PIC X(6)
00101	000990		VALUE SPACES.
00102	00100C	05 WS-NUMBER-DEPENDENTS	PIC Z9.
00103	001010	05 FILLER	PIC X(6)
00104	001020		VALUE IS SPACES.
00105	00103C	05 WS-PAY-AVERAGE	PIC \$\$\$9.99.

00106	001040	C5	FILLER	PIC X(2)
00107	001050			VALUE IS SPACES.
00108	001060	C5	WS-FEDERAL-TAX	PIC \$\$\$9.99.
00109	001070	05	FILLER	PIC X(5)
00110	001080			VALUE IS SPACES.
00111	001090	C5	WS-TAX-RATE	PIC V29.
00112	001100	05	FILLER	PIC X(7)
00113	001110			VALUE IS 'X'.
00114	001120	C5	WS-FICA-TAX	PIC \$\$\$9.99.
00115	001130	05	FILLER	PIC X(3)
00116	001140			VALUE IS SPACES.
00117	001150	C5	WS-NET-PAY	PIC \$\$\$9.99.
00118	001160	C5	FILLER	PIC X(11)
00119	001170			VALUE IS SPACES.
00120	001180	01	WS-TOTALING-LINE.	
00121	001190	05	FILLER	PIC X(35)
00122	001200			VALUE IS SPACES.
00123	001210	C5	FILLER	PIC X(31)
00124	001220			VALUE IS
00125	001230		'DEPARTMENT TOTAL FAMILY SIZE'.	
00126	001240	C5	WS-TOTAL-FAMILY-SIZE	PIC Z29.
00127	001250	C5	FILLER	PIC X(21)
00128	001260			VALUE IS 'AND TOTAL NET-PAY'.
00129	001270	C5	WS-TOTAL-NET-PAY	PIC \$\$\$9.99.
00130	001280	C5	FILLER	PIC X(34)
00131	001290			VALUE IS SPACES.
00132	001300		PROCEDURE DIVISION.	
00133	001310		OPEN INPUT PERSONNEL-MASTER-FILE	
00134	001320		OUTPUT PRINTER.	
00135	001330		MOVE WS-SWITCH-OFF TO WS-MASTER-FILE-EOF-SWITCH.	
00136	001340		PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT	
00137	001350		UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL TO WS-SWITCH-ON.	
00138	001360		CLOSE PERSONNEL-MASTER-FILE	
00139	001370		PRINTER.	
00140	001380		STOP RUN.	
00141	001390	0010	READ-AND-PRINT.	
00142	001400		READ PERSONNEL-MASTER-FILE	
00143	001410		AT END	
00144	001420		PERFORM 0020-TOTALS-RTN THRU 0020-EXIT	
00145	001430		MOVE WS-SWITCH-ON TO WS-MASTER-FILE-EOF-SWITCH	
00146	001440		GO TO 0010-EXIT.	
00147	001450		IF WS-DEPARTMENT-FLC EQUAL TO SPACES	
00148	001460		PERFORM 0030-HEADING-RTN THRU 0030-EXIT	
00149	001470		ELSE	
00150	001480		IF WS-DEPARTMENT-HOLD NOT EQUAL TO 1001-DEPARTMENT	
00151	001490		PERFORM 0020-TOTALS-RTN THRU 0020-EXIT	
00152	001500		PERFORM 0030-HEADING-RTN THRU 0030-EXIT.	
00153	001510		IF 1001-NUMBER-DEPENDENTS EQUAL TO ZEROS	
00154	001520		MOVE WS-3-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD	
00155	001530		ELSE	
00156	001540		IF 1001-NUMBER-DEPENDENTS LESS THAN 5	
00157	001550		MOVE WS-1-4-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD	
00158	001560		ELSE	

4

```

00159 001570      IF IM01-NUMBER-DEPENDENTS LESS THAN 4
00160 001580      MOVE WS-5-P-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00161 001590      ELSE
00162 001600      IF IM01-NUMBER-DEPENDENTS LESS THAN 13
00163 001610      MOVE WS-9-12-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00164 001620      ELSE
00165 001630      MOVE WS-13-99-DEPENDENTS-TAX-RATE TO
00166 001640      WS-TAX-RATE-HOLD.
00167 001650      MULTIPLY IM01-MONTHLY-PAY BY WS-TAX-RATE-HOLD
00168 001660      GIVING WS-FEDERAL-TAX-HOLD.
00169 001670      MULTIPLY IM01-MONTHLY-PAY BY WS-FICA-TAX-RATE
00170 001680      GIVING WS-FICA-TAX-HOLD.
00171 001690      SUBTRACT WS-FICA-TAX-HOLD WS-FEDERAL-TAX-HOLD FROM
00172 001700      IM01-MONTHLY-PAY
00173 001710      GIVING WS-NET-PAY-HOLD
00174 001720      DIVIDE WS-NET-PAY-HOLD BY WS-FAMILY-SIZE-HOLD
00175 001730      GIVING WS-PAY-AVERAGE.
00176 001740      MOVE IM01-SCC-SEC-NUM TO WS-SCC-SEC-NUM.
00177 001750      MOVE IM01-NAME TO WS-NAME.
00178 001760      MOVE IM01-MONTHLY-PAY TO WS-MONTHLY-PAY.
00179 001770      MOVE IM01-NUMBER-DEPENDENTS TO WS-NUMBER-DEPENDENTS.
00180 001780      MOVE WS-FEDERAL-TAX-HOLD TO WS-FEDERAL-TAX.
00181 001790      MOVE WS-TAX-RATE-HOLD TO WS-TAX-RATE.
00182 001800      MOVE WS-FICA-TAX-HOLD TO WS-FICA-TAX.
00183 001810      MOVE WS-NET-PAY-HOLD TO WS-NET-PAY.
00184 001820      ADD WS-NET-PAY-HOLD TO WS-TOTAL-NET-PAY-SUM.
00185 001830      ADD WS-FAMILY-SIZE-HOLD TO WS-TOTAL-FAMILY-SIZE-SUM.
00186 001840      WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
00187 001850      AFTER ADVANCING 2 LINES.
00188 001860      C01C-EXIT. EXIT.
00189 001870      C020-TOTALS-PTN.
00190 001880      MOVE WS-TOTAL-FAMILY-SIZE-SUM TO WS-TOTAL-FAMILY-SIZE.
00191 001890      MOVE WS-TOTAL-NET-PAY-SUM TO WS-TOTAL-NET-PAY.
00192 001900      WRITE OR01-PRINTER-RCD FROM WS-FOOTING-LINE
00193 001910      AFTER ADVANCING 3 LINES.
00194 001920      MOVE ZEROS TO WS-TOTAL-FAMILY-SIZE-SUM
00195 001930      WS-TOTAL-NET-PAY-SUM.
00196 001940      C02C-EXIT. EXIT.
00197 001950      C030-HEADING-RTN.
00198 001960      MOVE IM01-DEPARTMENT TO WS-DEPARTMENT-HOLD.
00199 001970      WRITE OR01-PRINTER-RCD FROM WS-COLUMN-HEADING
00200 001980      AFTER ADVANCING C*AN1.
00201 001990      C030-EXIT. EXIT.

```

INTRNL NAME	LVL	SOURCE NAME	BASE	DISPL	INTRNL NAME	DEFINITION	USAGE	R	O	O	M
DNM=1-151	FC	PERSONNEL-MASTER-FILE	DTF=01		DNM=1-151		DTFSD				F
DNM=1-156	01	IM01-PERSONNEL-MASTER-PCD	BL=1	000	DNM=1-156	DS 0CL80	GROUP				
DNM=1-234	02	IM01-SOC-SEC-NUM	BL=1	000	DNM=1-234	DS 9C	DISP				
DNM=1-260	02	IM01-NAME	BL=1	009	DNM=1-260	DS 18C	DISP				
DNM=1-279	02	IM01-DEPARTMENT	BL=1	018	DNM=1-279	DS 2C	DISP				
DNM=1-304	02	FILLER	BL=1	010	DNM=1-304	DS 5C	DISP				
DNM=1-320	02	IM01-NUMBER-DEPENDENTS	BL=1	022	DNM=1-320	DS 2C	DISP-NM				
DNM=1-352	02	IM01-MONTHLY-PAY	BL=1	024	DNM=1-352	DS 6C	DISP-NM				
DNM=1-378	02	FILLER	BL=1	024	DNM=1-378	DS 38C	DISP				
DNM=1-397	FD	PRINTER	DTF=02		DNM=1-397		DTFPR				F
DNM=1-428	01	OR01-PRINTER-ACC	BL=2	000	DNM=1-428	DS 133C	DISP				
DNM=1-454	77	WS-MASTER-FILE-ECF-SWITCH	BL=3	000	DNM=1-454	DS 1C	DISP				
DNM=2-000	77	WS-TOTAL-FAMILY-SIZE-SUM	BL=3	001	DNM=2-000	DS 2P	COMP-3				
DNM=2-034	77	WS-TOTAL-NET-PAY-SUM	BL=3	003	DNM=2-034	DS 4P	COMP-3				
DNM=2-067	77	WS-DEPARTMENT-PCD	BL=3	007	DNM=2-067	DS 2C	DISP				
DNM=2-095	77	WS-FEDERAL-TAX-HOLD	BL=3	009	DNM=2-095	DS 4P	COMP-3				
DNM=2-124	77	WS-FICA-TAX-PCD	BL=3	000	DNM=2-124	DS 4P	COMP-3				
DNM=2-150	77	WS-NET-PAY-HOLD	BL=3	011	DNM=2-150	DS 4P	COMP-3				
DNM=2-178	77	WS-FAMILY-SIZE-HOLD	BL=3	015	DNM=2-178	DS 2P	COMP-3				
DNM=2-207	77	WS-TAX-RATE-PCD	BL=3	017	DNM=2-207	DS 2P	COMP-3				
DNM=2-233	77	WS-FICA-TAX-RATE	BL=3	019	DNM=2-233	DS 2P	COMP-3				
DNM=2-259	77	WS-0-DEPENDENTS-TAX-RATE	BL=3	018	DNM=2-259	DS 2P	COMP-3				
DNM=2-293	77	WS-1-4-DEPENDENTS-TAX-RATE	BL=3	010	DNM=2-293	DS 2P	COMP-3				
DNM=2-329	77	WS-5-8-DEPENDENTS-TAX-RATE	BL=3	01F	DNM=2-329	DS 2P	COMP-3				
DNM=2-365	77	WS-9-12-DEPENDENTS-TAX-RATE	BL=3	021	DNM=2-365	DS 2P	COMP-3				
DNM=2-402	77	WS-13-99-DEPENDENTS-TAX-RATE	BL=3	023	DNM=2-402	DS 2P	COMP-3				
DNM=2-440	77	WS-SWITCH-ON	BL=3	025	DNM=2-440	DS 1C	DISP				
DNM=2-462	77	WS-SWITCH-OFF	BL=3	026	DNM=2-462	DS 1C	DISP				
DNM=3-000	01	WS-COLUMN-HEADING	BL=3	029	DNM=3-000	DS 0CL122	GROUP				
DNM=3-030	02	FILLER	BL=3	029	DNM=3-030	DS 14C	DISP				
DNM=3-049	02	FILLER	BL=3	036	DNM=3-049	DS 18C	DISP				
DNM=3-068	02	FILLER	BL=3	048	DNM=3-068	DS 15C	DISP				
DNM=3-087	02	FILLER	BL=3	057	DNM=3-087	DS 13C	DISP				
DNM=3-106	02	FILLER	BL=3	064	DNM=3-106	DS 11C	DISP				
DNM=3-125	02	FILLER	BL=3	06F	DNM=3-125	DS 10C	DISP				
DNM=3-144	02	FILLER	BL=3	079	DNM=3-144	DS 11C	DISP				
DNM=3-163	02	FILLER	BL=3	084	DNM=3-163	DS 11C	DISP				
DNM=3-182	02	FILLER	BL=3	08F	DNM=3-182	DS 19C	DISP				
DNM=3-201	01	WS-DETAIL-LINE	BL=3	048	DNM=3-201	DS 0CL133	GROUP				
DNM=3-231	02	FILLER	BL=3	048	DNM=3-231	DS 11C	DISP				
DNM=3-250	02	WS-SOC-SEC-NUM	BL=3	083	DNM=3-250	DS 9C	DISP				
DNM=3-274	02	FILLER	BL=3	08C	DNM=3-274	DS 5C	DISP				
DNM=3-293	02	WS-NAME	BL=3	0C1	DNM=3-293	DS 18C	DISP				
DNM=3-310	02	FILLER	BL=3	003	DNM=3-310	DS 6C	DISP				
DNM=3-329	02	WS-MONTHLY-PAY	BL=3	009	DNM=3-329	DS 8C	NM-EDIT				
DNM=3-364	02	FILLER	BL=3	0E1	DNM=3-364	DS 6C	DISP				
DNM=3-383	02	WS-NUMBER-DEPENDENTS	BL=3	0E7	DNM=3-383	DS 2C	NM-EDIT				
DNM=3-420	02	FILLER	BL=3	0E9	DNM=3-420	DS 6C	DISP				
DNM=3-439	02	WS-PAY-AVERAGE	BL=3	0EF	DNM=3-439	DS 8C	NM-EDIT				
DNM=3-474	02	FILLER	BL=3	0F7	DNM=3-474	DS 2C	DISP				

MEMORY MAP

TGT	00548
SAVE AREA	00549
SWITCH	00590
TALLY	00594
SCRT SAVE	00598
ENTPY-SAVE	0059C
SOPT CORE SIZE	005A0
ASTC-REELS	005A4
SORT RET	005A6
WORKING CELLS	005A8
SOPT FILE SIZE	00608
SORT MODE SIZE	0060C
PGT-VN TRL	006E0
TGT-VN TRL	006E4
SOPTAR ADDRESS	006F8
LENGTH OF VN TRL	006FC
LNGET OF SCRTAR	006FE
PGM ID	006F0
AI(INITI)	006F8
UPSI SWITCHES	006FC
OVERFLOW CELLS	007C4
RL CELLS	007C4
DTFACR CELLS	00710
TEMP STORAGE	00718
TEMP STORAGE-2	00728
TEMP STORAGE-3	00738
TEMP STORAGE-4	00738
BLL CELLS	00738
VLC CELLS	0073C
SRL CELLS	0073C
INDEX CELLS	0073C
SUBACR CELLS	0073C
ONCTL CELLS	0073C
PFMCTL CELLS	0073C
PFMSAV CELLS	0073C
VN CELLS	00750
SAVE AREA *2	0075C
XSASW CELLS	0075C
XSA CELLS	0075C
PARAM CELLS	0075C
PPTSAY AREA	0075C
CHECKPT CTR	0075C
TOPTR CELLS	0075C

LITERAL POOL (HEX)

007D0 (LIT+0)	0F5C9C01	3C402058	20202120	48202040	202120FC	20212000
007E8 (LIT+24)	CC0000C0	0C000004	5858C2C6	07C5C540	5858C2C3	D306E2C5

IA-02-02-22

PGT	0C768
OVERFLOW CELLS	0C768
VIRTUAL CELLS	00768
PROCEDURE NAME CELLS	0C76C
GENERATED NAME CELLS	0C784
SUBJECT ADDRESS CELLS	0C7C4
VNI CELLS	0C7C4
LITERALS	0C7C0
DISPLAY LITERALS	0C8C0

REGISTER ASSIGNMENT

REG 6 BL =3
 REG 7 BL =1
 REG 8 BL =2

133	0C0A00	START	EQU *	
	000A0C 5A 10 D 1CA		L 1,1CA(0,13)	DTF=1
	03C8C4 4B 10 C CE6		SH 1,086(0,12)	LIT+30
	0C0809 94 RF 1 000		NI 000(11),X'BF'	
	0CC80C 41 10 C C88		LA 1,088(0,12)	LIT+32
	3C0A10 58 00 D 1CA		L 0,1CA(0,13)	DTF=1
	CCC814 1A 40		LR 4,0	
	0C081A C5 F0		RALR 15,0	
	30081A 50 00 F CC8		ST 0,00A(0,15)	
	0CC81C 45 00 F 00C		BAL 0,00C(0,15)	
	30082C 00C00000		DC X'00000000'	
	00CA24 0A 02		SVC 2	
	0CC826 41 10 C 0A8		LA 1,08A(0,12)	LIT+32
	3CC82A 58 00 F 1CC		L 0,1CC(0,13)	DTF=2
	0C082E 1A 40		LR 4,0	
	000P30 07 00		PCR C,0	
	3CC832 C5 F0		RALP 15,0	
	30CA34 50 00 F CCP		ST 0,008(0,15)	
	300A3E 45 00 F 00C		BAL 0,00C(0,15)	
	0CC83C 0CC00000		DC X'00000000'	
	3CC840 0A 02		SVC 2	
	000842 50 20 D 1C0		ST 2,1C0(0,13)	BL =2
	33C846 58 A0 D 1C0		L 8,1C0(0,13)	BL =2
135	3C084A 02 00 6 CC0 6 026		MVC 000(1,6),026(6)	DNM=1-454
136	000P50 5A 00 D 2CA		L 0,208(0,13)	VN=01
	0CC854 50 00 C 1F4		ST 0,1F4(0,13)	PSV=1
	00CA58 58 00 C 01C		L 0,01C(0,12)	GN=01
	0CC85C 50 00 C 2CA		ST 0,208(0,13)	VN=01
	0CC86C	CA=01	EQU *	
	0CC86C 5A 20 C 020		L 2,020(0,12)	GN=02
	0C0864 C5 00 6 C00 6 025		CLC 000(1,6),025(6)	DNM=1-454
	000A6A 07 P2		BCR 8,2	DNM=2-440
	0CC86C 58 10 C 034		L 1,004(0,12)	PN=01

156	0009D4 07 F1	GA=011	HCR 15,1		
	0009D6		EQU *		
	JCO9D4 F2 71 D 1DC 7 022		PACK 1D0(8,13),022(2,7)	TS=01	DNM=1-320
	00C9DC F9 10 D 1D6 C 06A		CP 1D6(2,13),06A(1,12)	TS=07	LIT+1
	0009E2 5A F0 C 04C		L 15,04C(0,12)	GN=013	
	00C9E6 C7 AF		RCR 13,15		
157	JCC9EA F8 71 D 1DC 6 01D		ZAP 1D0(8,13),01D(2,6)	TS=01	DNM=2-293
	JCC9FE F1 76 D 1D0 C 100		MVFI 1D0(8,13),1D0(7,13)	TS=01	TS=01
	00C9F4 F8 11 6 017 D 1D4		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
159	0009FA 5A 10 C 048		L 1,048(0,12)	GN=012	
	JCO9FE 07 F1		RCR 15,1		
159	00CA0C	GA=013	EQU *		
	JCOA00 F2 71 D 1DC 7 022		PACK 1D0(8,13),022(2,7)	TS=01	DNM=1-320
	000A06 F9 10 D 1D6 C 06A		CP 1D6(2,13),06A(1,12)	TS=07	LIT+2
	JCOA0C 5A F0 C 050		L 15,050(0,12)	GN=014	
	00CA1C 07 AF		RCR 10,15		
160	JCOA12 F8 71 D 1DC 6 01F		ZAP 1D0(8,13),01F(2,6)	TS=01	DNM=2-329
	00CA18 F1 76 D 1D0 D 100		MVFI 1D0(8,13),1D0(7,13)	TS=01	TS=01
	JCOA1E F8 11 6 017 D 1D6		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
162	00CA24 5A 1C C 048		L 1,048(0,12)	GN=012	
	000A28 07 F1		RCR 15,1		
162	00CA2A	GN=014	EQU *		
	C0CA2A F2 71 D 1DC 7 022		PACK 1D0(8,13),022(2,7)	TS=01	DNM=1-320
	000A30 F9 11 D 1D6 C 06A		CP 1D6(2,13),06A(2,12)	TS=07	LIT+3
	000A36 5A F0 C 054		L 15,054(0,12)	GN=015	
	000A3A C7 AF		RCR 10,15		
163	000A3C F8 71 D 1DC 6 021		ZAP 1D0(8,13),021(2,6)	TS=01	DNM=2-365
	J0CA42 F1 76 D 1D0 D 100		MVFI 1D0(8,13),1D0(7,13)	TS=01	TS=01
	J0CA48 F8 11 6 017 D 1D6		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
165	000A4E 5A 10 C 048		L 1,048(0,12)	GN=012	
	J00A52 07 F1		RCR 15,1		
165	JCOA54	GA=015	EQU *		
	000A54 F8 71 D 1DC 6 023		ZAP 1D0(8,13),023(2,6)	TS=01	DNM=2-402
	000A5A F1 76 D 1D0 D 1C0		MVFI 1D0(8,13),1D0(7,13)	TS=01	TS=01
	000A60 F8 11 6 C17 D 1D6		ZAP 017(2,6),1D6(2,13)	DNM=2-207	TS=07
167	000A66	GN=012	EQU *		
	000A66 F2 75 D 1DC 7 024		PACK 1D0(8,13),024(6,7)	TS=01	DNM=1-352
	000A6C FC 51 D 1D2 6 017		VP 1D2(6,13),017(2,6)	TS=03	DNM=2-207
	00CA72 F1 76 D 1D0 D 100		MVFI 1D0(8,13),1D0(7,13)	TS=01	TS=01
	J00A78 F1 76 D 1D0 D 1C0		MVFI 1D0(8,13),1D0(7,13)	TS=01	TS=01
	000A7E F8 33 6 C09 D 1D4		ZAP 009(4,6),1D4(4,13)	DNM=2-95	TS=05
	000A84 94 0F 6 C09		NI 009(6),X'0F'	DNM=2-95	
169	000A88 F2 75 D 1DC 7 024		PACK 1D0(8,13),024(6,7)	TS=01	DNM=1-352
	000A8E FC 51 D 1D2 6 019		VP 1D2(6,13),019(2,6)	TS=03	DNM=2-233
	000A94 F1 75 D 1D0 D 1C0		MVFI 1D0(8,13),1D0(6,13)	TS=01	TS=01
	00CA9A F8 33 6 00C C 1D4		ZAP 00D(4,6),1D4(4,13)	DNM=2-124	TS=05
171	J00AAC F8 73 D 1DC 6 00D		ZAP 1D0(8,13),00D(4,6)	TS=01	DNM=2-124
	000AA6 FA 33 C 1D4 6 0C9		AP 1D4(4,13),009(4,6)	TS=05	DNM=2-95
	000AAC F2 75 D 1D8 7 024		PACK 1D8(8,13),024(6,7)	TS=09	DNM=1-352
	G00AB2 F8 33 C 1DC D 1D4		SP 1DC(4,13),1D4(4,13)	TS=013	TS=05
	J0CAB8 F8 33 6 011 D 1DC		ZAP 011(4,6),1DC(4,13)	DNM=2-150	TS=013
	000ARE 94 0F 6 011		NI 011(4),X'0F'	DNM=2-150	
174	000AC2 F8 73 D 1D8 6 C11		ZAP 1D8(8,13),011(4,6)	TS=09	DNM=2-150

	000ACA	FD 51 D 1CA 6 015	DP	1CA(6,13),015(2,6)	TS=011	DNM=2-178
	000ACE	F8 73 D 108 D 1CA	ZAP	1D8(8,13),1DA(4,13)	TS=09	TS=011
	000AD4	D2 09 D 1E0 C 06D	MVC	1E0(10,13),06D(12)	TS2=1	LIT+5
	000ACA	41 10 D 1E6	LA	1,1E6(0,13)	TS2=7	
	000ADE	DF 09 D 1E0 D 1DC	EDMK	1E0(10,13),1DC(13)	TS2=1	TS=013
	000AE4	C6 10	RCTR	1,0		
	000AE6	92 58 1 CCO	MVI	000(1),X'5B'		
176	00DAEA	D2 07 6 0EF 0 1E2	MVC	0EF(8,6),1E2(13)	DNM=3-439	TS2=3
177	000AF0	D2 08 6 083 7 000	MVC	083(9,6),000(7)	DNM=3-250	DNM=1-234
178	000AF6	D2 11 6 0C1 7 CC9	MVC	0C1(18,6),009(7)	DNM=3-293	DNM=1-260
	000AFC	F2 75 D 108 7 024	PACK	1D8(8,13),024(6,7)	TS=09	DNM=1-352
	000R02	D2 09 D 1E0 C 06D	MVC	1E0(10,13),06D(12)	TS2=1	LIT+5
	000RC8	41 10 D 1E6	LA	1,1E6(0,13)	TS2=7	
	000R0C	DF 09 D 1E0 D 1DC	EDMK	1E0(10,13),1DC(13)	TS2=1	TS=013
	000R12	06 10	RCTR	1,0		
	000R14	92 58 1 CCO	MVI	000(1),X'5B'		
179	000R18	D2 07 6 0D9 0 1E2	MVC	0D9(9,6),1E2(13)	DNM=3-329	TS2=3
	000R1E	F2 71 D 108 7 022	PACK	1D8(8,13),022(2,7)	TS=09	DNM=1-320
	000R24	D2 03 D 1EC C 077	MVC	1E0(4,13),077(12)	TS2=1	LIT+5
	000R2A	DE 03 D 1E0 C 10E	ED	1E0(4,13),10E(13)	TS2=1	TS=015
	000R30	D2 01 6 0F7 D 1E2	MVC	0F7(2,6),1E2(13)	DNM=3-383	TS2=3
180	000R36	D2 09 D 1E0 C 06D	MVC	1E0(10,13),06D(12)	TS2=1	LIT+5
	000R3C	41 10 D 1E6	LA	1,1E6(0,13)	TS2=7	
	000R40	DF 09 D 1EC 6 CC9	EDMK	1E0(10,13),009(6)	TS2=1	DNM=2-95
	000R46	C6 10	RCTR	1,0		
	000R48	92 58 1 000	MVI	000(1),X'5B'		
181	000R4C	D2 07 6 0F9 D 1E2	MVC	0F9(8,6),1E2(13)	DNM=4-0	TS2=3
	000R52	F8 71 D 108 6 017	ZAP	1D8(8,13),017(2,6)	TS=09	DNM=2-207
	000R58	D7 05 D 108 C 108	XC	1D8(6,13),1D8(13)	TS=09	TS=09
	000R5E	94 0F D 1DE	NI	1DE(13),X'0F'	TS=09+6	
	000R62	D2 03 D 1EC C 078	MVC	1E0(4,13),078(12)	TS2=1	LIT+19
	000R68	DE 03 D 1E0 C 10E	ED	1E0(4,13),10E(13)	TS2=1	TS=015
	000R6E	D2 01 6 1C6 D 1E2	MVC	106(2,6),1E2(13)	DNM=4-54	TS2=3
182	000R74	D2 09 D 1E0 C 06D	MVC	1E0(10,13),06D(12)	TS2=1	LIT+5
	000R7A	41 10 D 1E6	LA	1,1E6(0,13)	TS2=7	
	000R7E	DF 09 C 1EC 6 00D	EDMK	1E0(10,13),00D(6)	TS2=1	DNM=2-124
	000R84	C6 10	RCTR	1,0		
	000R86	92 58 1 CCO	MVI	000(1),X'5B'		
183	000R8A	D2 07 6 10F 0 1E2	MVC	10F(8,6),1E2(13)	DNM=4-99	TS2=3
	000R9C	D2 09 D 1E0 C 06D	MVC	1E0(10,13),06D(12)	TS2=1	LIT+5
	000R96	41 10 C 1E6	LA	1,1E6(0,13)	TS2=7	
	000R9A	DF 09 D 1EC 6 011	EDMK	1E0(10,13),011(6)	TS2=1	DNM=2-150
	000RA0	06 10	RCTR	1,0		
	000RA2	92 58 1 CCO	MVI	000(1),X'5B'		
184	000RA6	D2 07 6 11A D 1E2	MVC	11A(8,6),1E2(13)	DNM=4-150	TS2=3
185	000RAC	FA 33 6 003 6 011	AP	003(4,6),011(4,6)	DNM=2-34	DNM=2-150
186	000RA2	FA 11 6 001 6 015	AP	001(2,6),015(2,6)	DNM=2-0	DNM=2-178
	000R88	D2 84 8 00C 6 0A8	MVC	000(133,8),0A8(6)	DNM=1-428	DNM=3-201
	000R8E	58 10 C 1CC	L	1,1CC(0,13)	DTF=2	
	000R02	92 FC 8 CCO	MVI	000(9),X'F0'	DNM=1-428	
	000R06	58 F0 1 010	L	15,010(0,1)		
	000RCA	45 E0 F C0C	RAL	14,00C(0,15)		
	000RCE	50 20 D 1C0	ST	2,1C0(0,13)	BL =2	

CPCSS-REFERENCE DICTIONARY

DATA NAMES	DEFN	REFERENCE
PERSONNEL-MASTER-FILE	00012	00133 00133 00138 00142 00142
IM01-SOC-SEC-NUM	00021	00176
IM01-NAME	00022	00177
IM01-DEPARTMENT	00023	00150 00198
IM01-NUMBER-DEPENDENTS	00025	00153 00156 00159 00162 00179
IM01-MONTHLY-PAY	00026	00167 00169 00171 00178
PRINTER	00014	00133 00133 00138 00186 00192 00199
OR01-PRINTER-RCO	00030	00186 00186 00186 00192 00192 00192 00192 00199 00199 00199 00199 00199
WS-MASTER-FILE-ECF-SWITCH	00032	00135 00136 00145
WS-TOTAL-FAMILY-SIZE-SUM	00033	00185 00190 00194
WS-TOTAL-NET-PAY-SUM	00035	00184 00191 00194
WS-DEPARTMENT-HOLD	00037	00147 00147 00147 00150 00198
WS-FEDERAL-TAX-HOLD	00039	00167 00167 00171 00180
WS-FICA-TAX-HOLD	00041	00169 00171 00182
WS-NET-PAY-HOLD	00043	00171 00171 00174 00183 00184
WS-FAMILY-SIZE-HOLD	00045	00174 00185
WS-TAX-RATE-HOLD	00047	00154 00157 00160 00163 00165 00167 00181
WS-FICA-TAX-RATE	00049	00169
WS-0-DEPENDENTS-TAX-RATE	00052	00154
WS-1-4-DEPENDENTS-TAX-RATE	00055	00157
WS-5-8-DEPENDENTS-TAX-RATE	00058	00160
WS-9-12-DEPENDENTS-TAX-RATE	00061	00163
WS-13-99-DEPENDENTS-TAX-RATE	00064	00165
WS-SWITCH-CN	00067	00136 00145
WS-SWITCH-CFF	00069	00135
WS-COLUMN-HEADING	00071	00199 00199
WS-DETAIL-LINE	00090	00186 00186
WS-SOC-SEC-NUM	00093	00176
WS-NAME	00096	00177
WS-MONTHLY-PAY	00099	00178
WS-NUMBER-DEPENDENTS	00102	00179
WS-PAY-AVERAGE	00105	00174
WS-FEDERAL-TAX	00108	00180
WS-TAX-RATE	00111	00181
WS-FICA-TAX	00114	00182
WS-NET-PAY	00117	00183
WS-FOOTING-LINE	00120	00192 00197
WS-TOTAL-FAMILY-SIZE	00126	00190
WS-TOTAL-NET-PAY	00129	00191

PROCEDURE NAMES	DEFN	REFERENCE
0010-READ-AND-PRINT	00141	00136
0010-EXIT	00188	00136 00146
0020-TOTALS-RTN	00189	00144 00151
0020-EXIT	00196	00144 00151

IA-02-02-22

28/08/78	PHASE	XFR-AD	LCCCPH	PICCPH	CSK-AC	ESD TYPE	LABEL	LOADED	REL-FR
	PHASE***	CC5000	005000	00A213	2E 12 4	CSECT	DUMPOPE	005000	005000
						CSECT	IJGFIEWZ	005E28	005E28
						* ENTRY	IJGFIZ47	005E28	
						* ENTRY	IJGFIZZZ	005E28	
						* ENTRY	IJGFIEZZ	005E28	
						CSECT	ILRDSAEO	006000	006000
						ENTRY	ILRDSAE1	0060F0	
						CSECT	IJDFAPIZ	005D10	005D10
						* ENTRY	IJDFAZIZ	005D10	
						CSECT	ILBDMNSO	0060C8	0060C8
						WXTRN	STXITPSW		
						WXTRN	ILRDBG2		

• UNREFERENCED SYMBOLS

002 UNRESOLVED ADDRESS CONSTANTS

SSN

NAME

MONTHLY PAY NO OF DEPS PAY AVE

FED TAX

FICA TAX

NET-PAY

1A-02-02-22

13

301 285

PAGE 1

.....

.....	\$38EQJ3A.....	B
...&.....	6R.....	K
.....4.....U.....	
\$262FBH6.....	
...C...O.....A...O....	
.....	...&.....	
.....-...S...A...../...	
.....B...O....	O
...O...9...P.....	
.....8...Z.....&.....	
...K.....K.....	...7...&K.....	
...O.....O.....	...O.....	
.....O...K.....	...-.....F...	
...O...K.....	...O.....	
...FK...B.....	O...K.....	
.....	
.....O.....	
.....	...O...K.....H...	
...K.....O.....	
...../* /&.....	O.....	
.....	
.....KN.....&.....	
.....O.....	...-...-.....-...	
...&-.....H...-...	&-...D...&.....	
K.....9...-	...-...&.....&...	
...O...28/08/78	&&.....	
DUMPE2...P.....+	
.....I.....*280878240	
.....-.....	*.....\$180.....	
.....	...4.....	
.....	
.....	
...O...O...B9.....	
.....B...&...	
...K...H...&...U...	...K...A.....	
9&.....K.....	...D...K.....9	
K.....&.....O.....	
.....K.....	
.....&.....	...-...-...-...	
-.....-&-...	...H...-&-...	
...D...&.....	9...&.....-	
...-.....&.....9.....	

```

00004E8C 006337C9 37003991 39923A5C
C00036A8 00000014 19201996 1A3C1A4C
CCCC00C0 3C0400CC 00000000 03680080
C0001A74 0000000C 00000000 C0000000
CCCC0000 0000000C 000005A0 00000030
CC001A74 0000000C 00000C00 00000000
CCCC0000 00000000 000005A0 00000128

C709F000 E01F000F C402F013 0130130F
C402F313 3133133F C402F413 4134134F
C402F714 2142142F E307F018 0180180F
E3C7F318 3183183F E3D7F418 4184184F
C5D5C44C 40000000 C5D5C440 40004500

CCCC0001 00000006 06040009 40404040

```

```

.....+ .....I.....+
.....2A0878240 .....
+. ....
.....
.....
.....
.....
.....
.....
....
CPO.....CPO..... PRO.....OKO.....
DK1.....DK2..... DK3.....DK4.....
DK5.....DK6..... DK7.....TP0.....
TP1.....TP2..... TP3.....TP4.....
TP5.....TD0..... END ..END ...
....
$RQUMPB$BDUMPB .....Q

```

FF150007	E00054CE	00005C9E	00005000
C0C05A54	00005C82	00005A66	000054C0
00005308	000054C0	00000000	01767F86
CC0C5000	000107FF	00005000	0001D784
0A0407F1	00005010	00005010	00005FA8
7341733E	4740703A	FA1C734A	73CF9F11
7187728A	02087192	7260D232	71A37299
95C0E000	4770F0A2	9610D048	92FFFE00
FC9A9110	00480719	C7FF070C	00005C9E
CC005800	00005C84	C306C2C6	F0F0F0F1
F1000454	4C000329	4C004896	2CE24000
40404040	40404040	40404040	4040E2E2
D5C104C5	40404040	40404040	40404040
C640C4C5	C7E240D7	C1E840C1	E5C54040
40E3C1E7	404040C5	C5E360D7	C1E84040
40404040	40404040	404040C5	E340D7C1
4C404040	40404040	40404040	40404040
4C6C4040	404040D9	4C404040	40404040
4C4C4040	404040C8	40404040	40404040
4C4C4040	40404040	404040C4	C507C1D9
C3E840F2	C9E9C540	4040C4C5	D740C1D5
4040D3E8	40E2C9E9	C5404040	40404040
CCCC0000	00000000	00008234	0C00C106
E2F0F0F6	40010000	00000000	00028020
CCCC0000	00460000	02005904	00C80013
CC0053D8	00000050	00005427	8E0060D0
C8005328	00C00000	06D05380	00000050
CC005370	00000537	00005D10	088489F1
01005431	20000084	06005418	00000050
C2EE4C40	40404040	404040F0	F1F0F1D7
40404040	40404040	40404040	40404040
0C1C58E0	F06407FE	F7F9F4F7	F4F4F0F2

NO NAME
.....
.....*.....
.....Q.....
.....P.....P.....
.....E.....E.....
.....K.....9.....
.....K.....K.....
.....O.....0.....
.....O.....*.....
.....*CORFOOO1
.....S.....
SS
NAME
MONTHLY PAY NO O F DEPS PAY AVE
FED TAX FICA TAX NET-PAY
TE F ET PA
Y PA

R
H
DEPAR
LY SIZE DEP AN
LY SIZE

SY S006
Q.....H.....
-0.....-.....
24
A.....
639167039HOTT RO RY 0101P
FC09039800
.....Q.....784764GZ

005860 D1F0C078 DE03C1F0 0103D201
005880 D1E060C0 C6109258 1000L207 610F01F2
0058AC C610C528 10C0D7C7 611AF1F2 FA3360C3
0058C0 D1C092F0 PC0058F0 101045F0 FC0C5020
0058FC C077C0F0 C1E060C0 02026177 01F1F873
0058CC C1E0C0C0 4110D1E6 PCF9C1F0 010CC0A1
0058C0 924089C4 5810D1CC 924C80C0 58FC1010
0058C4 C07FD2C3 60C03C81 5810C20C 07F1F201
0058C6 C079807A 591071CC 92F1E30C 58FC1010
0058C0 C7F10ACE 5C0050C8 5C50C304 582CC0C0
0058AC C05435FC 9120D048 47F0F01A 58C0C48
0058CC 0004411C C0044170 C068C670 05505840
0058C0 4173C1CF C010580C 8CCC1E08 5CC8C0C3
005800 D18C5880 D1C058F0 C05407FE 7CC096F0
005820 C9D1C4C6 C1C7C9F9 F3F5918C 10024710
0058C4 1017FC00 19CC180C 43C01017 41F0C010
005860 43DFE107 43C01016 19CD4770 F0769180
005880 4713FC76 0AC742D0 1C2842D0 1C16CAC3
0058AC 92011C28 58E01028 02021029 10195C00
0058CC 0A03C7FE 51801002 4710F08E 0AC79101
0058E0 47F0F00A 91C11027 078E1200 4780F0F4
0058C0 C0005548 FCC05C74 F2C2C1F9 F9F7F6F5
0058C2 A398F388 C2181308 47F0F014 1F0F252
0058C4 C7C6C9C5 E6F9F3F9 0CC890AE F26C9180
005860 104A50A0 1C609143 10274710 F1CC47F0
00588C 1015478C FC7E1800 41ACC109 1FCA47C0
0058A0 00FF44C0 F25A98AE F26C4400 105407FE
0058CC 910A1064 4710F0A2 0A0C9180 10024710
0058EC 910R1003 4780F0C4 91801064 4780F0D0
0058FC 91801064 4780F14C 58F01064 988DF26C
005820 0A0CF26C 47F0F154 CCCC0000 CCCC0000
0058F0 48810300 4980F260 4780F13C 4880105E
005860 47F0F158 91041064 4780F0C8 58E010C0
00588C 588010C8 4188C0C0 5C801058 914C1049
0058A0 48C0104A 48C10300 1AC850C0 10609740
0058CC 000518CC 43C81038 43F81063 19CD4770
0058F0 000142C8 10380503 103C1036 47D0F202
006000 4110F264 0AC29801 F28C9120 10104710
00602C F0AC91C8 1C494710 F2149108 10644780
006040 0A039110 10494780 F22A5710 104947F0
00606C 1026471C F1CC9120 1C274710 F1CC94FE
006080 C7FED20C DCC09C00 CCCC0001 5858C20F
0060A0 00005548 800058F6 C0300000 CCCC0000
0060CC CCCCC000 C000C7FE FFC81045 4780F0C8
0060E0 F4F3F0F3 F1F3F761 F7F41800 47F0F022
0061CC C0CC9110 40144780 F03C91C8 40154710
006120 F03C5811 C0044154 C0C04850 F11A5855
00614C F0E43000 4780F05A 41330C02 47F0F0C8
00616C C0C644C3 C0CA1852 92F05030 02065C01
006180 96015C02 47F0F09E 96015001 47F0F0C8
0061A0 07FF89C0 C0184840 F11C1604 4110F10A
0061C0 F116C7C1 1CC21002 4111C38C 5CD10CC0
0061F0 43091C43 19CD4770 F18F42F8 1C3888F0
006200 5858C2C3 F0F2C5D9 C0C0C000 C0C00007

106F1E2 0209D1F0 C0A04110 0126C039
C2C901F0 C0A04110 01E60F09 01F06011
6011F411 6001601F 02848030 60A85810
D1C05880 01C05810 020407F1 0203C1F0
C1C96033 0703D108 0108943F 01C00209
92581CC0 0207418A 01E20293 83006130
45F0F0C0 5020D1C0 5880D1C0 02016001
C0C77018 02798000 60289240 837AD209
45F0F0C0 5020D1C0 5880D1C0 5810D210
95C07030 077922FF 20039610 DC4850E0
98208C5C 58E0D054 07FE9620 00484160
10031E48 50401000 87165030 4180D19C
57861C00 02040208 C05C5860 01C45870
0A320000 47F0F084 04320030 47F0F01A
F0240A37 90CFF0FC 58E01018 06E00200
43CEFCF7 190C4780 F05346E0 F3400A32
10164790 F0769208 10280A30 91801002
91801002 4710F08A 0A07D201 1C261003
1C180A50 98C0F0EC 44031022 58E0F0F4
1C154790 F0D494FE 10159132 1026C78E
18F007FF 92881028 0A0007FE 03005768
F4F3C3F1 4E60F040 9380D3C8 C3888388
47F0F0C2 47F0F014 47F0F014 0A32C901
10494710 F04A58A0 102C50A0 10584AA0
FCAC9880 1058968C F0885030 10589110
FC76D2FF 00303030 1A0A8684 F06441CC
91081049 4780F098 96F71049 47F0F1CC
F0AC0A07 91C11004 4780F088 47F0F236
58E01064 47F0F0F4 91921033 4780F110
9CF1F100 58101090 070005EE 98F1E00C
CCCC0000 C0000000 91401035 4780F158
4880F260 4720F124 4740F13C 96401049
47FCFC84 91201064 4780F158 96101049
4710F178 4AB0104A 50801060 47F0F18A
104948E0 103489F0 000843E0 F2634180
F18647E8 103888EC C0084680 F19C41CC
96C81049 9001F280 98A0F28C 4130F284
FC7E9140 1027478C F1FA949F 102747F0
F21A5880 1080D202 10811020 5080102C
FC888680 10499880 105847F0 F0569180
10499640 102747FC F1CCD203 10581060
C7C5F540 00005000 C0005030 03005768
FFCC0000 00000000 00000030 00000000
C5F047F0 F018C903 C2C4E2C1 C5F0F3F1
41C0C001 05F050E0 F0E59035 F0F24144
F0309120 40027480 FC2C4100 0C1047F0
C0C04155 00031255 4790F0AC 1835D501
5C50F0FA 05033006 F0F04780 F09E4403
5C001200 4780F09A 4900F056 4780F092
56C15030 58E0F0CF 9805F0F2 58F0F0EA
0A02910F 00484780 F03C5910 F11258F0
C7FF1800 0A060CC0 000005FC C0011038
CCCC4880 F19C41CC 003142C9 1C38D503
00340008 00000030 00000300 00000300

J.K.J.....J.J.K. /JSK.J.....JW..
J.....S.K./JS K.J.....JH..J..
...S.K./JS.. -...-...K...-...
J...O...O...O..f. J...J...K...K.J..
...J...K.../J..8. JC...P.JQJQ..J.K..
J.....JW...J.J... ..S.K./JSK.../..
...J...-...O... ..O...J...J.K...-..
..K...-...K...K... -...K...-... ..K..
.....J...l...O... ..O...J...J...K..
l...E..E..E..... ..E.....E..
...O.....O..... ..E.....E.....-..
.....E..... ..E.....E.....J..
..J.....E..... ..K.K...-JD..
J...J.....O..... ..O.....O.....O..
IJDFAPI239..... ..O.....O.....K..
..... ..07.....0E...0..
..l.....O..... ..O.....O.....
..O..... ..O.....O.....K.....
.....K.....E..... ..O.....O.....04
.....O..... ..O.....O.....
..O..... ..O.....O.....
.....*28A98765 43C1+-O ..L.C..
..T.....O.....02. ..C0...O0...O0...IJ
GFIEWZ39....2... ..O.....E.....
..E...-... ..l...0 ..O.....O...E.....
...O..... ..O.K.....O.....
...2...2..... ..O.....O.....7...01..
.....O..... ..O.....O.....O.....02..
.....O0.....O..... ..O.....O0U.....1..
.....l.....2..... ..l.....l.....
..2...01..... ..O..... ..l..
.....2...l..... ..2...l... l... ..
..01.....O0...E ..O0U.....l.....
.....E..... ..l.....E...-01..
.....A.....E...- ..O.....2...
.....N.....2..... ..O.....2...2...2..
..2.....2..... ..O... ..l.....0..
O.....2..... ..2.....K.....E..
.....2..... ..O.....O.....O0..
...l.....l..... ..O... ..01.K.....-
..K.....*8BD PEN ..E...E.....
.....6..... ..O.....
.....O..... ..O...O0..ILB0SAE031
4303/07/74...00.O...O...02..
... ..O... ..O... ..O.....O..
O.....E1... ..O.....O.....N..
O11...O.....O0. ..E0.N...O...O..
.....O...K...E. ..E.....O...O0...O..
..E...O0...E...O0. ..E...O...02...O0..
.....l.....l..... ..O.....O...l...O..
l.P.....EJ... ..O.....O.....
.....l..... ..l.....N..
*8AC048..... ..O.....

DEBUG

USAIA

SOFTWARE DIVISION, CSD

PRACTICAL EXERCISE NUMBER 4

PROGRAM NAME: DUMP-PE

IA-02-02-14

1

306


```

CRL L19
00001      BASIS C320MP
00002 00001C IDENTIFICATION DIVISION.
00003 000020 PROGRAM-ID. CUMF-PE.
00004 00003C AUTHOR. LT MUDGIN.
00005 000040 REMARKS. PE DESIGNED FOR USING ALC TO DERUG A COROL PROGRAM.
00006 00005C ENVIRONMENT DIVISION.
00007 00006C CONFIGURATION SECTION.
00008 000070 SPECIAL-NAMES.
00009 00008C      COL IS CHAN1.
00010 00009C INFL-OUTPUT SECTION.
00011 00010C FILE-CONTROL.
00012 00011C      SELECT PERSONNEL-MASTER-FILE
00013 000120      ASSIGN TO SYS006-UT-2314-S.
00014 00013C      SELECT PRINTER
00015 000140      ASSIGN TO SYS005-UP-1403-S.
00016 00015C DATA DIVISION.
00017 00016C FILE SECTION.
00018 00017C FO PERSONNEL-MASTER-FILE
00019 00018C LABEL RECORDS STANDARD.
00020 00019C 01 IMOI-PERSONNEL-MASTER-RCO.
00021 00020C      C5 IMCI-SFC-SEC-NUM          PIC X(9).
00022 00021C      C5 IMCI-NAME                  PIC X(19).
00023 00022C      C5 IMOI-DEPARTMENT          PIC X(2).
00024 00023C      C5 FILLER                    PIC X(5).
00025 00024C      C5 IMOI-NUMBER-DEPENDENTS  PIC 9(2).
00026 00025C      C5 IMCI-MONTHLY-PAY        PIC 9(4)V9(2).
00027 00026C      C5 FILLER                    PIC X(38).
00028 00027C FO PRINTER
00029 00028C LABEL RECORDS OMITTED.
00030 00029C 01 OPOI-PRINTER-RCC              PIC X(133).
00031 00030C WORKING-STORE SECTION.
00032 00031C 77 WS-MASTER-FILE-EOF-SWITCH  PIC X.
00033 00032C 77 WS-TOTAL-FAMILY-SIZE-SUM    PIC S9(3)
00034 I      VALUE IS ZERO
00035 00033C      CC0330      LSAGE IS COMP-3.
00036 00034C 77 WS-TOTAL-NET-PAY-SUM        PIC S9(5)V9(2)
00037 I      VALUE IS ZERO
00038 00035C      CC0350      LSAGE IS COMP-3.
00039 00036C 77 WS-DEPARTMENT-HOLD          PIC X(2)
00040 00037C      VALUE IS SPACES.
00041 00038C 77 WS-FEDERAL-TAX-HOLD        PIC S9(4)V9(2)
00042 00039C      LSAGE IS COMP-3.
00043 00040C 77 WS-FICA-TAX-HOLD           PIC S9(4)V9(2)
00044 00041C      LSAGE IS COMP-3.
00045 00042C 77 WS-NET-PAY-HOLD            PIC S9(4)V9(2)
00046 00043C      LSAGE IS COMP-3.
00047 00044C 77 WS-FAMILY-SIZE-HOLD       PIC S9(3)
00048 00045C      LSAGE IS COMP-3.
00049 00046C 77 WS-TAX-RATE-HOLD          PIC S9V9(2)
00050 00047C      LSAGE IS COMP-3.
00051 00048C 77 WS-FICA-TAX-RATE          PIC S9(3)
00052 I      VALUE IS +.059

```

00053	00049C		LSAGE IS COMP-3.
00054	00050C	77 WS-0-DEPENDENTS-TAX-RATE	PIC SV9(3)
00055	000510		LSAGE IS COMP-3
00056	000520		VALUE IS +.2.
00057	000530	77 WS-1-4-DEPENDENTS-TAX-RATE	PIC SV9(3)
00058	000540		LSAGE IS COMP-3
00059	000550		VALUE IS +.15.
00060	00056C	77 WS-5-8-DEPENDENTS-TAX-RATE	PIC SV9(3)
00061	000570		LSAGE IS COMP-3
00062	000580		VALUE IS +.12.
00063	00059C	77 WS-9-12-DEPENDENTS-TAX-RATE	PIC SV9(3)
00064	00060C		LSAGE IS COMP-3
00065	000610		VALUE IS +.09.
00066	00062C	77 WS-13-59-DEPENDENTS-TAX-RATE	PIC SV9(3)
00067	000630		LSAGE IS COMP-3
00068	00064C		VALUE IS +.05.
00069	000650	77 WS-SWITCH-CA	PIC X
00070	00066C		VALUE IS '0'.
00071	00067C	77 WS-SWITCH-OFF	PIC X
00072	000680		VALUE IS '1'.
00073	00069C	01 WS-COLUMN-HEADING.	
00074	00070C	C5 FILLER	PIC X(14)
00075	00071C		VALUE IS SPACES.
00076	00072C	05 FILLER	PIC X(14)
00077	000730		VALUE IS 'SSN'.
00078	00074C	05 FILLER	PIC X(15)
00079	000750		VALUE IS 'NAME'.
00080	000760	05 FILLER	PIC X(13)
00081	000770		VALUE IS 'MONTHLY PAY'.
00082	000780	05 FILLER	PIC X(11)
00083	00079C		VALUE IS 'NO OF DEPS'.
00084	00080C	05 FILLER	PIC X(11)
00085	00081C		VALUE IS 'PAY AVE'.
00086	00082C	05 FILLER	PIC X(10)
00087	000830		VALUE IS 'FED TAX'.
00088	00084C	05 FILLER	PIC X(12)
00089	000850		VALUE IS 'TAX RATE'.
00090	00086C	C5 FILLER	PIC X(11)
00091	00087C		VALUE 'FICA TAX'.
00092	00088C	C5 FILLER	PIC X(12)
00093	00089C		VALUE 'NET PAY'.
00094	00090C	01 WS-DETAIL-LINE.	
00095	00091C	C5 FILLER	PIC X(11)
00096	000920		VALUE SPACES.
00097	000930	05 WS-SEC-SEC-NUM	PIC X(9).
00098	00094C	C5 FILLER	PIC X(5)
00099	000950		VALUE IS SPACES.
00100	00096C	C5 WS-NAME	PIC X(18).
00101	00097C	C5 FILLER	PIC X(6)
00102	00098C		VALUE SPACES.
00103	00099C	05 WS-MONTHLY-PAY	PIC \$\$\$9.99.
00104	00100C	05 FILLER	PIC X(6)
00105	00101C		VALUE SPACES.

IA-02-02-14

00106	C0100C	05	WS-NUMBER-DEPENDENTS	PIC Z9.
00107	001010	05	FILLER	PIC X(6)
00108	001020			VALUE IS SPACES.
00109	00103C	05	WS-PAY-AVERAGE	PIC \$\$\$9.99.
00110	00104C	05	FILLER	PIC X(2)
00111	00105C			VALUE IS SPACES.
00112	00106C	05	WS-FEDERAL-TAX	PIC \$\$\$9.99.
00113	00107C	05	FILLER	PIC X(5)
00114	00108C			VALUE IS SPACES.
00115	00109C	05	WS-TAX-RATE	PIC VZ9.
00116	00110C	05	FILLER	PIC X(7)
00117	00111C			VALUE IS 'X'.
00118	00112C	05	WS-FICA-TAX	PIC \$\$\$9.99.
00119	001130	05	FILLER	PIC X(3)
00120	00114C			VALUE IS SPACES.
00121	001150	05	WS-NET-PAY	PIC \$\$\$9.99.
00122	001160	05	FILLER	PIC X(11)
00123	00117C			VALUE IS SPACES.
00124	001180	01	WS-FOOTING-LINE.	
00125	001190	05	FILLER	PIC X(35)
00126	00120C			VALUE IS SPACES.
00127	001210	05	FILLER	PIC X(31)
00128	00122C			VALUE IS
00129	001230		'DEPARTMENT TOTAL FAMILY SIZE'.	
00130	001240	05	WS-TOTAL-FAMILY-SIZE	PIC ZZ9.
00131	00125C	05	FILLER	PIC X(21)
00132	001260			VALUE IS ' AND TOTAL NET PAY'.
00133	00127C	05	WS-TOTAL-NET-PAY	PIC \$\$\$9.99.
00134	00128C	05	FILLER	PIC X(34)
00135	001290			VALUE IS SPACES.
00136	00130C		PROCEDURE DIVISION.	
00137	001310		OPEN INPUT PERSONNEL-MASTER-FILE	
00138	00132C		OUTPUT PRINTER.	
00139	00133C		MOVE WS-SWITCH-OFF TO WS-MASTER-FILE-EOF-SWITCH.	
00140	00134C		PERFORM 001C-READ-AND-PRINT THRU 0010-EXIT	
00141	00135C		UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL TO WS-SWITCH-ON.	
00142	00136C		CLOSE PERSONNEL-MASTER-FILE	
00143	00137C		PRINTER.	
00144	001380		STOP RUN.	
00145	00139C		001C-READ-AND-PRINT.	
00146	001400		READ PERSONNEL-MASTER-FILE	
00147	00141C		AT END	
00148	001420		PERFORM 002C-TOTALS-RTN THRU 0020-EXIT	
00149	001430		MOVE WS-SWITCH-ON TO WS-MASTER-FILE-EOF-SWITCH	
00150	00144C		GO TO 0010-EXIT.	
00151	001450		IF WS-DEPARTMENT-HOLD EQUAL TO SPACES	
00152	001460		PERFORM 0030-HEADING-RTN THRU 0030-EXIT	
00153	00147C		ELSE	
00154	001480		IF WS-DEPARTMENT-HOLD NOT EQUAL TO 1401-DEPARTMENT	
00155	00149C		PERFORM 0020-TOTALS-RTN THRU 0020-EXIT	
00156	001500		PERFORM 0030-HEADING-RTN THRU 0030-EXIT.	
00157	00151C		IF 1401-NUMBER-DEPENDENTS EQUAL TO ZEROS	
00158	00152C		MOVE WS-0-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD	

```

00159 001530 ELSE
00160 001540 IF IM01-NUMBER-DEPENDENTS LESS THAN 5
00161 001550 MOVE WS-1-4-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00162 001560 ELSE
00163 001570 IF IM01-NUMBER-DEPENDENTS LESS THAN 9
00164 001580 MOVE WS-5-8-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00165 001590 ELSE
00166 001600 IF IM01-NUMBER-DEPENDENTS LESS THAN 13
00167 001610 MOVE WS-9-12-DEPENDENTS-TAX-RATE TO WS-TAX-RATE-HOLD
00168 001620 ELSE
00169 001630 MOVE WS-13-99-DEPENDENTS-TAX-RATE TO
00170 001640 WS-TAX-RATE-HOLD.
00171 1 ADD 1 IM01-NUMBER-DEPENDENTS
00172 1 GIVING WS-FAMILY-SIZE-HOLD.
00173 001650 MULTIPLY IM01-MONTHLY-PAY BY WS-TAX-RATE-HOLD
00174 001660 GIVING WS-FEDERAL-TAX-HOLD.
00175 001670 MULTIPLY IM01-MONTHLY-PAY BY WS-FICA-TAX-RATE
00176 001680 GIVING WS-FICA-TAX-HOLD.
00177 001690 SUBTRACT WS-FICA-TAX-HOLD WS-FEDERAL-TAX-HOLD FROM
00178 001700 IM01-MONTHLY-PAY
00179 001710 GIVING WS-NET-PAY-HOLD
00180 001720 DIVIDE WS-NET-PAY-HOLD BY WS-FAMILY-SIZE-HOLD
00181 001730 GIVING WS-PAY-AVERAGE.
00182 001740 MOVE IM01-SCC-SEC-NO TO WS-SCC-SEC-NO.
00183 001750 MOVE IM01-NAME TO WS-NAME.
00184 001760 MOVE IM01-MONTHLY-PAY TO WS-MONTHLY-PAY.
00185 001770 MOVE IM01-NUMBER-DEPENDENTS TO WS-NUMBER-DEPENDENTS.
00186 001780 MOVE WS-FEDERAL-TAX-HOLD TO WS-FEDERAL-TAX.
00187 001790 MOVE WS-TAX-RATE-HOLD TO WS-TAX-RATE.
00188 001800 MOVE WS-FICA-TAX-HOLD TO WS-FICA-TAX.
00189 001810 MOVE WS-NET-PAY-HOLD TO WS-NET-PAY.
00190 001820 ADD WS-NET-PAY-HOLD TO WS-TOTAL-NET-PAY-SUM.
00191 001830 ADD WS-FAMILY-SIZE-HOLD TO WS-TOTAL-FAMILY-SIZE-SUM.
00192 001840 WRITE CR01-PRINTER-RCD FROM WS-DETAIL-LINE
00193 001850 AFTER ADVANCING 2 LINES.
00194 001860 COIC-EXIT. EXIT.
00195 001870 CO20-TOTALS-RTN.
00196 001880 MOVE WS-TOTAL-FAMILY-SIZE-SUM TO WS-TOTAL-FAMILY-SIZE.
00197 001890 MOVE WS-TOTAL-NET-PAY-SUM TO WS-TOTAL-NET-PAY.
00198 001900 WRITE CR01-PRINTER-RCD FROM WS-FOOTING-LINE
00199 001910 AFTER ADVANCING 3 LINES.
00200 001920 MOVE ZEROS TO WS-TOTAL-FAMILY-SIZE-SUM
00201 001930 WS-TOTAL-NET-PAY-SUM.
00202 001940 CO20-EXIT. EXIT.
00203 001950 CO20-HEADING-RTN.
00204 001960 MOVE IM01-DEPARTMENT TO WS-DEPARTMENT-HOLD.
00205 001970 WRITE CR01-PRINTER-RCD FROM WS-COLUMN-HEADING
00206 001980 AFTER ADVANCING CHAN1.
00207 001990 CO20-EXIT. EXIT.

```

INTRNL NAME	LVL	SOURCE NAME	BASE	DISPL	INTRNL NAME	DEFINITION	USAGE	R	O	Q
DNM=1-151	FD	PERSONNEL-MASTER-FILE	DTF=01		DNM=1-151		DTFSD			
DNM=1-156	01	IM01-PERSONNEL-MASTER-RCO	BL=1	000	DNM=1-196	DS 0CL80	GROUP			
DNM=1-234	02	IM01-SOC-SEC-NUM	BL=1	000	DNM=1-234	DS 9C	DISP			
DNM=1-260	02	IM01-NAME	BL=1	009	DNM=1-260	DS 18C	DISP			
DNM=1-279	02	IM01-DEPARTMENT	BL=1	018	DNM=1-279	DS 2C	DISP			
DNM=1-304	02	FILLER	BL=1	010	DNM=1-304	DS 5C	DISP			
DNM=1-320	02	IM01-NUMBER-DEPENDENTS	BL=1	022	DNM=1-320	DS 2C	DISP-NM			
DNM=1-352	02	IM01-MONTHLY-PAY	BL=1	024	DNM=1-352	DS 6C	DISP-NM			
DNM=1-378	02	FILLER	BL=1	02A	DNM=1-378	DS 3AC	DISP			
DNM=1-397	FD	PRINTER	DTF=02		DNM=1-397		DTFPR			
DNM=1-428	01	OR01-PRINTER-RCO	BL=2	000	DNM=1-428	DS 133C	DISP			
DNM=1-454	77	WS-MASTER-FILE-EOF-SWITCH	BL=3	000	DNM=1-454	DS 1C	DISP			
DNM=2-000	77	WS-TOTAL-FAMILY-SIZE-SUM	BL=3	001	DNM=2-000	DS 2P	COMP-3			
DNM=2-034	77	WS-TOTAL-NET-PAY-SUM	BL=3	003	DNM=2-034	DS 4P	COMP-3			
DNM=2-067	77	WS-DEPARTMENT-FOLD	BL=3	007	DNM=2-067	DS 2C	DISP			
DNM=2-095	77	WS-FEDERAL-TAX-HOLD	BL=3	009	DNM=2-095	DS 4P	COMP-3			
DNM=2-124	77	WS-FICA-TAX-HOLD	BL=3	000	DNM=2-124	DS 4P	COMP-3			
DNM=2-150	77	WS-NET-PAY-HOLD	BL=3	011	DNM=2-150	DS 4P	COMP-3			
DNM=2-178	77	WS-FAMILY-SIZE-HOLD	BL=3	015	DNM=2-178	DS 2P	COMP-3			
DNM=2-207	77	WS-TAX-RATE-HOLD	BL=3	017	DNM=2-207	DS 2P	COMP-3			
DNM=2-233	77	WS-FICA-TAX-RATE	BL=3	019	DNM=2-233	DS 2P	COMP-3			
DNM=2-259	77	WS-0-DEPENDENTS-TAX-RATE	BL=3	018	DNM=2-259	DS 2P	COMP-3			
DNM=2-293	77	WS-1-4-DEPENDENTS-TAX-RATE	BL=3	010	DNM=2-293	DS 2P	COMP-3			
DNM=2-329	77	WS-5-9-DEPENDENTS-TAX-RATE	BL=3	01F	DNM=2-329	DS 2P	COMP-3			
DNM=2-365	77	WS-9-12-DEPENDENTS-TAX-RATE	BL=3	021	DNM=2-365	DS 2P	COMP-3			
DNM=2-402	77	WS-13-99-DEPENDENTS-TAX-RATE	BL=3	023	DNM=2-402	DS 2P	COMP-3			
DNM=2-440	77	WS-SWITCH-ON	BL=3	025	DNM=2-440	DS 1C	DISP			
DNM=2-462	77	WS-SWITCH-OFF	BL=3	026	DNM=2-462	DS 1C	DISP			
DNM=3-000	01	WS-COLUMN-HEADING	BL=3	028	DNM=3-000	DS 0CL127	GROUP			
DNM=3-030	02	FILLER	BL=3	028	DNM=3-030	DS 14C	DISP			
DNM=3-049	02	FILLER	BL=3	036	DNM=3-049	DS 18C	DISP			
DNM=3-068	02	FILLER	BL=3	048	DNM=3-068	DS 15C	DISP			
DNM=3-087	02	FILLER	BL=3	057	DNM=3-087	DS 13C	DISP			
DNM=3-106	02	FILLER	BL=3	064	DNM=3-106	DS 11C	DISP			
DNM=3-125	02	FILLER	BL=3	06F	DNM=3-125	DS 11C	DISP			
DNM=3-144	02	FILLER	BL=3	07A	DNM=3-144	DS 10C	DISP			
DNM=3-163	02	FILLER	BL=3	084	DNM=3-163	DS 12C	DISP			
DNM=3-182	02	FILLER	BL=3	090	DNM=3-182	DS 11C	DISP			
DNM=3-201	02	FILLER	BL=3	098	DNM=3-201	DS 12C	DISP			
DNM=3-220	01	WS-DETAIL-LINE	BL=3	0A8	DNM=3-220	DS 0CL133	GROUP			
DNM=3-250	02	FILLER	BL=3	0A8	DNM=3-250	DS 11C	DISP			
DNM=3-269	02	WS-SOC-SEC-NUM	BL=3	083	DNM=3-269	DS 9C	DISP			
DNM=3-293	02	FILLER	BL=3	08C	DNM=3-293	DS 5C	DISP			
DNM=3-312	02	WS-NAME	BL=3	0C1	DNM=3-312	DS 18C	DISP			
DNM=3-329	02	FILLER	BL=3	003	DNM=3-329	DS 6C	DISP			
DNM=3-348	02	WS-MONTHLY-PAY	BL=3	009	DNM=3-348	DS 8C	NM-EDIT			
DNM=3-383	02	FILLER	BL=3	0E1	DNM=3-383	DS 6C	DISP			
DNM=3-402	02	WS-NUMBER-DEPENDENTS	BL=3	0E7	DNM=3-402	DS 2C	NM-EDIT			
DNM=3-439	02	FILLER	BL=3	0E9	DNM=3-439	DS 6C	DISP			
DNM=3-458	02	WS-PAY-AVEFACE	BL=3	0EF	DNM=3-458	DS 8C	NM-EDIT			

MEMORY MAP

TGT	00549
SAVE AREA	00549
SWITCH	00550
TALLY	00594
SOFT SAVE	00559
ENTRY-SAVE	00550
SOFT CORE SIZE	005A0
ASTC-REELS	005A4
SOFT RET	005A6
WORKING CELLS	005A9
SOFT FILE SIZE	00608
SOFT MODE SIZE	0060C
PGT-VN TBL	006F0
TGT-VN TBL	006E4
SORTAB ADDRESS	006EF
LENGTH OF VN TBL	006EC
LNGETH OF SORTAB	006EE
PGM ID	006F0
A(INIT1)	006F8
UPSI SWITCHES	006FC
OVERFLOW CELLS	00704
BL CELLS	00704
DTFACR CELLS	00710
TEMP STORAGE	00718
TEMP STORAGE-2	00728
TEMP STORAGE-3	00738
TEMP STORAGE-4	00738
RLI CELLS	00738
VLC CELLS	0073C
SRL CELLS	0073C
INDEX CELLS	0073C
SIMPACR CELLS	0073C
CNCTL CELLS	0073C
PFMCTL CELLS	0073C
PFMSAV CELLS	0073C
VN CELLS	00750
SAVE AREA #2	00750
XSASW CELLS	00750
XSA CELLS	00750
PARAM CELLS	00750
RPTSAV AREA	00750
CHFCMPT CTR	00750
ICPTP CELLS	00750

LITERAL POOL (HEX)

00700 (LIT+0)	CF5C9CC1	3C1C4020	58202021	2C4A2020	40202120	F3202120
00708 (LIT+24)	000CC0C0	0C0C0004	5858C2D6	C7C5D540	5858C2C3	D3D6E2C5

IA-02-02-14

PGT	OC76A
OVERFLOW CELLS	CC768
VIRTUAL CELLS	OC769
PROCEDURE NAME CELLS	OC76C
GENERATED NAME CELLS	CC784
SUPPOT ADDRESS CELLS	OC7C4
VMI CELLS	OC7C4
LITERALS	OC7C0
DISPLAY LITERALS	CCAC0

REGISTER ASSIGNMENT

REG 6 RL =3
 REG 7 RL =1
 REG 8 RL =2

137	CCC800	START	EQU	*	
	CCC80C 58 10 0 1C8		L	1,1C8(0,13)	DTF=1
	CCC804 4A 10 C C66		SH	1,086(0,12)	LIT+30
	CCC808 94 AF 1 C00		NI	000(1),X*AF	
	CCC80C 41 10 C 0AA		LA	1,088(0,12)	LIT+32
	CCC810 58 00 D 1C8		L	0,1C8(0,13)	DTF=1
	CCC814 1A 40		LR	4,0	
	CCC816 C5 F0		BALR	15,0	
	CCC81A 50 00 F CCA		ST	0,008(0,15)	
	CCC81C 45 00 F 00C		BAL	0,00C(0,15)	
	CCC82C C0C0C0C0		DC	X'00000000'	
	CCC824 0A 02		SVC	2	
	CCC826 41 10 C 0AA		LA	1,088(0,12)	LIT+32
	CCC82A 58 00 D 1CC		L	0,1CC(0,13)	DTF=2
	CCC82E 1A 40		LR	4,0	
	CCC83C C7 00		BCR	0,0	
	CCC832 C5 F0		BALR	15,0	
	CCC834 50 00 F CCA		ST	0,008(0,15)	
	CCC83A 45 00 F 00C		BAL	0,00C(0,15)	
	CCC83C C0C0C0C0		DC	X'00000000'	
	CCC840 0A 02		SVC	2	
	CCC842 50 20 C 1C0		ST	2,1C0(0,13)	RL =2
	CCC846 58 80 C 1C0		L	8,1C0(0,13)	RL =2
139	CCC84A 02 00 6 C00 6 026		MVC	000(1,6),026(6)	DNM=1-454
140	CCC850 58 00 D 2CA		L	0,208(0,13)	VN=01
	CCC854 5C 00 D 1F4		ST	0,1F4(0,13)	PSV=1
	CCC85E 58 00 C 01C		L	0,01C(0,12)	GN=01
	CCC85C 50 00 C 209		ST	0,208(0,13)	VN=01
	CCC86C	GA=01	EQU	*	
	CCC860 5A 20 C 020		L	2,020(0,12)	GN=02
	CCC864 05 00 6 C00 6 025		CLC	000(1,6),025(6)	DNM=1-454
	CCC86A C7 A2		PCR	A,2	DNM=2-440
	CCC86C 58 10 C 004		L	1,004(0,12)	PN=01

	00091A	58 00 0 1FA		L	0,1FA(0,13)	PSV=2	
	00091C	50 00 0 20C		ST	0,20C(0,13)	VN=02	
149	000922	02 00 6 000 6 025		MVC	000(1,6),025(6)	DNM=1-454	CNM=2-440
150	00092A	58 10 0 00A		L	1,00A(0,12)	PN=02	
	00092C	07 F1		ACP	15,1		
151	00092F		GN=04	ECU	*		
	00092F	58 20 0 030		L	2,030(0,12)	GN=06	
	000932	55 40 6 007		CLC	007(6),X*40	DNM=2-67	
	000936	07 72		ACP	7,2		
	00093A	05 00 6 008 6 007		CLC	008(1,6),007(6)	DNM=2-67+1	CNM=2-67
	00093E	07 72		BCR	7,2		
152	000940	58 00 0 210		L	0,210(0,13)	VN=03	
	000944	50 00 0 1FC		ST	0,1FC(0,13)	PSV=3	
	000948	58 00 0 034		L	0,034(0,12)	GN=07	
	00094C	50 00 0 210		ST	0,210(0,13)	VN=03	
	000950	58 10 0 010		L	1,010(0,12)	PN=04	
	000954	07 F1		ACP	15,1		
	000956		GN=07	ECU	*		
	000956	58 00 0 1FC		L	0,1FC(0,13)	PSV=3	
	00095A	50 00 0 210		ST	0,210(0,13)	VN=03	
154	00095E	58 10 0 038		L	1,038(0,12)	GN=08	
	000962	07 F1		ACP	15,1		
154	000964		GN=06	ECU	*		
	000964	58 20 0 038		L	2,038(0,12)	GN=08	
	000968	05 01 6 007 7 01A		CLC	007(2,6),018(7)	DNM=2-67	DNM=1-279
	00096E	07 82		BCR	8,2		
155	000970	58 00 0 20C		L	0,20C(0,13)	VN=02	
	000974	50 00 0 200		ST	0,200(0,13)	PSV=4	
	000978	58 00 0 030		L	0,030(0,12)	GN=09	
	00097C	50 00 0 20C		ST	0,20C(0,13)	VN=02	
	000980	58 10 0 000		L	1,000(0,12)	PN=03	
	000984	07 F1		BCR	15,1		
	000986		GN=05	ECU	*		
	000986	58 00 0 200		L	0,200(0,13)	PSV=4	
	00098A	50 00 0 20C		ST	0,20C(0,13)	VN=02	
156	00098E	58 00 0 210		L	0,210(0,13)	VN=03	
	000992	50 00 0 204		ST	0,204(0,13)	PSV=5	
	000996	58 00 0 040		L	0,040(0,12)	GN=010	
	00099A	50 00 0 210		ST	0,210(0,13)	VN=03	
	00099E	58 10 0 010		L	1,010(0,12)	PN=04	
	0009A2	07 F1		ACP	15,1		
	0009A4		GN=010	ECU	*		
	0009A4	58 00 0 204		L	0,204(0,13)	PSV=5	
	0009A8	50 00 0 210		ST	0,210(0,13)	VN=03	
157	0009AC		GN=08	ECU	*		
	0009AC	F2 71 0 100 7 022		PACK	100(8,13),022(2,7)	TS=01	DNM=1-320
	0009A2	F9 10 0 106 0 068		CP	106(2,13),068(1,12)	TS=07	LIT=0
	000988	58 F0 0 044		L	15,044(0,12)	GN=011	
	00098C	07 7F		ACP	7,15		
158	00098E	F8 71 0 100 6 018		ZAP	100(8,13),018(2,6)	TS=01	DNM=2-259
	0009C4	F1 76 0 100 0 100		MVC	100(8,13),100(7,13)	TS=01	TS=01
	0009CA	F8 11 6 017 0 10A		ZAP	017(2,6),106(2,13)	DNM=2-207	TS=07
160	0009DC	58 10 0 048		L	1,048(0,12)	GN=012	

IA-02-02-14

CROSS-REFERENCE DICTIONARY

DATA NAMES	DEFN	REFERENCE
PERSONNEL-MASTER-FILE	00012	00137 00137 00142 00146 00146
INC1-SOC-SEC-NUM	00021	00182
INC1-NAME	00022	00183
INC1-DEPARTMENT	00023	00134 00204
INC1-NUMBER-DEPENDENTS	00025	00157 00160 00163 00166 00171 00185
INC1-MONTHLY-PAY	00026	00173 00175 00177 00184
PRINTER	00014	00137 00137 00142 00192 00198 00205
PRO1-PRINTER-RCO	00030	00192 00192 00192 00198 00198 00198 00198 00205 00205 00205 00205 00205
WS-MASTER-FILE-ECF-SWITCH	00032	00139 00140 00149
WS-TOTAL-FAMILY-SIZE-SUM	00033	00191 00196 00200
WS-TOTAL-NET-PAY-SUM	00036	00190 00197 00200
WS-DEPARTMENT-FCLO	00039	00151 00151 00151 00154 00204
WS-FEDERAL-TAX-HOLD	00041	00173 00173 00177 00186
WS-FICA-TAX-HOLD	00043	00175 00177 00189
WS-NET-PAY-HOLD	00045	00177 00177 00180 00199 00190
WS-FAMILY-SIZE-HOLD	00047	00171 00180 00191
WS-TAX-FATE-HOLD	00049	00158 00161 00164 00167 00169 00173 00187
WS-FICA-TAX-RATE	00051	00175
WS-0-DEPENDENTS-TAX-RATE	00054	00159
WS-1-4-DEPENDENTS-TAX-RATE	00057	00161
WS-5-9-DEPENDENTS-TAX-RATE	00060	00164
WS-9-12-DEPENDENTS-TAX-RATE	00063	00167
WS-13-99-DEPENDENTS-TAX-RATE	00066	00169
WS-SWITCH-ON	00069	00140 00149
WS-SWITCH-OFF	00071	00139
WS-COLUMN-HEADING	00073	00205 00205
WS-DETAIL-LINE	00094	00192 00192
WS-SOC-SEC-NUM	00057	00182
WS-NAME	00100	00183
WS-MONTHLY-PAY	00103	00184
WS-NUMBER-DEPENDENTS	00106	00185
WS-PAY-AVERAGE	00109	00180
WS-FEDERAL-TAX	00112	00186
WS-TAX-RATE	00115	00187
WS-FICA-TAX	00118	00188
WS-NET-PAY	00121	00189
WS-FOOTING-LINE	00124	00199
WS-TOTAL-FAMILY-SIZE	00130	00196
WS-TOTAL-NET-PAY	00133	00197

PROCEDURE NAMES	DEFN	REFERENCE
0010-READ-ANC-PRJAT	00145	CC14C
0010-EXIT	00194	CC140 00150
0020-TOTALS-RTN	CC195	CC148 00155
0020-EXIT	00202	CC148 00155

28/09/79	PHASE	XFR-AD	LOCORE	MICORE	DSK-AC	ESD TYPE	LABEL	LOADED	REL-FR
	PHASE***	005000	005000	005000	2E 12 4	CSECT	005000	005000	
						CSECT	IJGF15WZ	005E3A	005E3A
						* ENTRY	IJGF12WZ	005E3A	
						* ENTRY	IJGF12ZZ	005E3A	
						* ENTRY	IJGF1EZZ	005E3A	
						CSECT	ILRDSAE0	0060E0	0060E0
						ENTRY	ILRDSAE1	006100	
						CSECT	IJDFAPIZ	005020	005020
						* ENTRY	IJDFAZIZ	005020	
						CSECT	ILPD4NS0	006008	006008
						WXTRN	STXITPSW		
						WXTRN	IL8008G2		

* UNREFERENCED SYMBOLS

002 UNRESOLVED ADDRESS CONSTANTS

SSN	NAME	MONTHLY PAY	NO OF DEPS	PAY AVE	FED TAX	TAX RATE	FICA TAX	NET PAY
784764025	GRUNDY GARVEY JR	\$569.00	12	\$37.66	\$45.44	08 %	\$32.94	\$489.62
639167039	HOTT RORY	\$398.00	9	\$34.30	\$31.84	08 %	\$23.08	\$343.08
257808345	CHANEY HELEN WILLI	\$344.00	8	\$31.41	\$41.28	12 %	\$19.95	\$282.77

SSN	NAME	MONTHLY PAY	NO OF DEPS	PAY AVE	FED TAX	TAX RATE	FICA TAX	NET PAY
535411P04	RITCHISON PENNIE L	\$568.00	0	\$421.46	\$113.60	20 %	\$32.94	\$421.46

DEPARTMENT TOTAL FAMILY SIZE 32 AND TOTAL NET PAY \$1115.47

DEPARTMENT TOTAL FAMILY SIZE 1 AND TOTAL NET PAY \$421.46

SSN

NAME

MONTHLY PAY NO OF DEPS PAY AVE

FED TAX

TAX RATE

FICA TAX

NET PAY

05031 PROGRAM CHECK INTERRUPTON - HEX LOCATION 0C5902 - CONDITION CODE 3 - DATA EXCEPTION
05001 JOB DUMPED CANCELED

IA-02-02-14

13

GR C-7 00005C64 000059A4 000054C0 00005000
GR R-F 000054C0 000054C0 00005000 00005000
C049EC 00 0000 15 00036A

CC0C52PA 50005CF0 0000505A 000053DA
CC0C576P 0000554A R0005C96 00005020

030C00	0C7C0C00	CC00000J	00004C6A	CC000438
030C2J	FF050300	43C04H8A	FF150007	FOC0598A
0CC04C	CC0C1E5C	CC0C0C0C	CC0C1B2F	CCCCC070
03004C	0C04CC0C	CC037B54	CC04C30C	CC002397
0CC0FC	59F7F6F2	C6C2C8F6	CCCCC0C0	CC0C0003
0000A3	457C0148	540FC393	41A0D658	457C81P0
0300C0	068CC6A0	C6P00680	068FC680	41P80017
0300E0	9120A0C0	471000FC	9260A001	95E2A002
03010J	47800110	51C4A000	478CC11C	9203008A
0C012J	41AAE64F	C77874F9	7038C701	7C587058
00014C	A001457C	CC5A07F8	4283C0F5	588CC283
0C0160	902C0218	90100298	02C79008	8CC05680
CC018C	C2C4912C	P77A4780	018F5285	C6A8CC07
0C01A3	42A33397	41AA7648	023CC589	03979540
0C01CC	C58C47F0	C1CC0700	C58C588A	459002CA
0C01EC	471CC8C6	C2070258	9C0848A0	03C241AA
0C0200	459C02CA	4400A004	5890A004	98C8902C
0C022J	P20C0020	52F406A8	91E0877A	47800262
0C0240	9103877P	47700178	9503C093	4780C178
0C026C	025802C7	02588A98	989C8A10	P2C00298
00C280	56010C38	P2000038	03615C40	61504C44
0C02A0	00C38FA8	50038F94	00037088	03038FA0
0302CJ	0CC03CC0	CC004000	36L2C500	C589058A
0C02F0	02E8A87C	CC0847F0	02EE8870	CC081F78
0C030C	477C074C	12774740	03401866	4A700580
03032C	8C2C5C60	8C205880	05C81867	5F6C8C20
0C0340	P200058A	05899868	05ACC7F9	18664360
0C036C	801C47FC	C3400000	F2F861F0	F861F7F8
0C0380	C4F4B47C	77C3F340	0001C7FF	0CC06223
0003A3	28AC7EC0	CC6337C9	370CC991	39923A5C
0003CC	000C3F48	CC030014	1F601594	1A3C1A4C
0C03F0	010C1P04	36C40000	0000C000	03681080
00040C	00001A74	CC0C0C00	00C0C000	CCCCC000
0C042C	0C0C0C00	CC004500	CC0CC5A0	00003F38
0C044J	588E0288	47F0061C	47F00646	4AA003C2
0C046C	5880A0C0	9CC8802C	588EC288	5C68C5A0
0C0480	02889C69	C5A04590	04F09869	05A055FF
0C04AC	547F877A	C2188AC8	02985300	8AF448F0
0C04CJ	07F95030	888C48E0	0772C207	8A88FCC0
0004EC	02C0C089	058A9878	0C5C5970	P22447D0
0C050J	18971C78	5E70058C	9180A000	4780051E
0C0520	058A4770	C5729140	058AC719	5080058C
00054J	40600582	588005CC	1F615F6C	80245C60
0C056C	5880C5C4	5E7C8024	507C8024	9240C58A
0C0580	058A8F6C	CC025888	05C0557C	8C1C5070
0C05A0	000021FE	90001C18	0C00CC02	800005E4
0005C0	C0004258	CC003F38	0CC0403C	CC004129
0C05F0	459C06C8	41900200	5800C023	478CC8C2
CC060J	02C84E66	7C7007F6	F2E5C3F1	F0F0956A

CC0C0010	000J036R	FF05J030	00000000
5R5RC2C5	D6D1F3C1	FF350130	80035E5A
FA20EC00	026E29FC	0C040300	0F002302
CCCC0JC	000J11E4	0C040000	000386A4
CC05C003	000336RC	36R00690	419R0073
1RAA4190	017R41R0	C1A347F0	00DC0680
41PRC050	4570014R	41R0017R	9640A001
47RC944A	95C1A002	47R0R44A	9561A002
92R1AC0C	4RA003C2	487A0604	49A003D0
92R3A000	9680A0C1	440009EA	078R547E
61R0A000	07175R9C	A0J44R90	02225000
AC0C07F7	45700150	D207R432	8A3R58E0
FA32C648	1RAA0D0R	RA3200JC	43A10009
C5R94740	01CC9560	058947R0	01C69240
47FC06A2	58R0A004	4220A000	9140A001
6C640701	0J16A000	9R93901J	8200029R
91020021	47R0021C	920005RA	9R9F029R
9120R77A	47100254	91R0R86A	4710C254
47F00262	0207C25R	RAC0R9R0	RACR8200
96R0A000	41100030	47F0R8FE	96030039
FCC0C0C0	00000000	FF050133	80004R84
CC034CC4	80001DAE	00001000	00002000
C7R9906R	05A0987R	00505970	8A2447D0
1FF71C7R	5E7005RC	50900590	9130058A
5C6005R4	406005R0	58R005C0	1R675E60
5C6C0020	58R005C4	5E70R020	50708020
05RAP660	00025R86	05C05E70	801C5070
5CC05000	00000000	00000000	00000000
CCC06223	0000001C	0003FFFF	F8F04E80
3C703C74	3C7R38F2	F8F3F8F7	F8F2F4F0
1A5CC070	378C0010	5858C2D6	00060007
000003F4	03000000	0000387C	00000044
CCCC0010	00000000	00000000	00000000
CCCC0C0C	00CC0000	923R0223	909FC29R
55RRA000	47800468	91R0A000	4710047A
45RC04E0	9R6R05A0	07F996J1	A0009RRE
A00F07R9	91A0R77A	47R004C2	9640R77A
C222D207	RAC0E000	94F0R8AC1	9R9R02RR
5RE002C4	94FD8A59	021RRA40	029R07FR
0AFAR870	000R47F0	04FER870	00081F78
55FFAC0F	47R0051E	D200059R	059A9130
12774740	056C1R66	4A7005A2	5D60C584
8C745R80	05C81R67	5E60R024	50608024
C7F9957R	02230785	5090059C	18664360
EC1C57C2	02000000	9240059A	C7F90000
CC0C0000	00000001	101J1000	F87F559R
CCCC0000	00000050	909F027R	92200223
5535C023	47R006CE	4R600022	1A664R70
00234770	00C09200	002307F1	41900D80

[illegible]

```

0043AC 0007A06C CCC00030 0003FFFF 0AFD4580
0043CD 3C7C2C74 2C7A7AF2 FAF0F8F7 FPF2F4F0
0043FC 1A5CCC2C 778C0C3C CCCCC000 000CCCC0
004403 00004444 C0C0C000 00003F7C 00000344
00442C CCCCCCCC C0C0C0C0 C0C0C000 C0C0C0C0
004440 CCCCC00C C0C0C000 00003F7C 00000344
004460 C0C0C0C0 00000000 00000000 C0C0C0C0
00448C CCCCCCCC --SAME--
004500 C3F9F3C0 C0C000CF C3D7F300 DC18000F
004520 C4D2F113 1131111F C4D2F213 2132127F
004540 C4D2F514 014014CF C4D2F614 1141141F
004560 E3D7F118 1181181F F3D7F218 2182182F
004580 F3D7F518 5185185F F3C4F018 C180180F
0045AC C0C0C0C0 --SAME--
004760 5853C2C4 E4D4C7C2 5854C2C4 F4C4C7C2
00478C 40404C40 --SAME--
00480C 404C4040 4C404040 404C4040 40404040

```

```

20C34E85 306337C9 37333491 3492345C
C0033648 00030014 19201496 1A3C144C
CC030300 3604000C 30030000 03680080
CC031A74 00030300 0C030303 30000000
CCC05C0C 0003000C 00300540 0C004030
CC031A74 30030300 00030003 03030300
CCC0C0C0 0C0000C0 07000540 03004128

C709FC00 E01E000F C402F013 0130130F
C402F313 3133133F C402F413 4134134F
C4C2F714 2142142F E307F018 3180180F
F307F318 3183183F E307F418 4184184F
F505F440 40000300 C505C440 40004500

CC000301 00030306 C6040338 46404040

```

```
. . . . . +      ..+.....#  
. . . . . 280878240 .....*  
.*.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
  
....  
CRO...CPQ.....PRO...PKO..  
DK1...DK2.....DK3...DK4..  
DK5..DK6.....DK7...TPQ..  
TP1...TP2.....TP3...TP4..  
TP5...TDQ.....END   END..  
  
....  
$$$RUMPR$$RCMPB    ....Q
```

LALTYP FEX LENGTH IS CCQC

- 86 -

004FA0				D5C640D5	C1D4C540
004FCC	C0C050C0	C0005768	CC005548	8CC0C0FA	
004FEC	C0C050C0	C000528A	50005CE0	000C50E8	
005000	05FC07C0	530EF0D0	47FCF082	0CC050C0	
0C5C20	FFFFFFF7C	C0005000	00004F8A	0CC0C870	
00504C	00006FA8	CC000368	D202733E	7299F912	
0050A0	734A7348	47807014	D2117168	7276D202	
0C5080	028A728A	58C0F0C6	58E0C0C0	58D0F0CA	
0050A0	47FCF0AC	58CEFF0A	9CECC0C0	18F0589F	
0050C0	00005000	00005000	00005768	0C005548	
0C50EC	C4E404D7	F0D7C540	F1CC0C00	CC0CCCF0	
0C5100	0C058C20	CC150C12	0CC8C0C5	0CF0F1F0	
0C5120	D54C4C40	40404C4C	4C404040	40404040	
005140	C6D5E3C8	D3E840D7	C1E84040	D5D64C06	
005160	4040C6C5	C440E3C1	E7404040	E3C1E740	
00518C	4C40C405	C5E34C07	C1E8404C	404040C6	
0051A0	F1F8F0F4	4C404C40	40C9C9E3	C3C8C9E2	
0C51C0	404058F5	F8F84FF0	F0404040	40404040	
0051E0	404058F1	F1F348FA	F04C4040	404CF2F0	
00520C	40404058	F4F2F148	F4F64040	40404C40	
0C522C	404C4C4C	40404040	40404040	404C4C40	
005240	F3C4C505	E340F3D6	E3C1C340	C6C1D4C9	
0C5260	C44CE3D6	F3C1D340	D5C5E340	C7C1E840	
005280	404C4040	--SAME--			
0C52AC	C0700C50	C0C00000	G00C0000	0CC00000	
0052CC	C00C532C	C1C05340	04305F38	200EE2E8	
0052E0	00C0C0C0	C40053D8	80CC0C00	CC000C46	
005300	288C0CCF	CC460000	01CC610C	5821CC58	
005320	C70C52F2	4C0C0C0C	313052F4	4CC0CC05	
005340	28000CCF	CC000000	0CC0C000	0CC00105	
0C5360	010054C1	CC000000	C70C4120	FCC0000C	
0C538C	F5F3F5F4	F1F1F8F0	F4D9C9F3	C3C8C5F2	
0C53A0	C6C3F0F0	FCF5F6F8	F0FC4040	404C4C40	
0053C0	4C404040	40404040	404C4C4C	4C404040	

FF150007	F003598A	00005C90	00005000
CCCC5D2C	00005C94	00005944	000054C3
000053D8	000054C3	00003100	01775529
CCCC5C00	0001D7FF	00005100	00010784
CAC4C7F1	00005010	0C005010	00005FA8
7341733F	4740703A	FA107348	73CF9911
7167724A	D2087192	7260C202	71A37299
9500E030	4770F0A2	9E10DC8A	92FFFE00
FCBA9110	00480719	07DF0700	00005C80
CC005A00	00005C96	306AC2C6	F0F0F0F1
F3001136	0C003329	4C304214	6C001C02
4C4C4040	40404040	40404040	4040E2E2
5C1D4C5	40404040	40404040	404040D4
C44C4C5	07E240D7	C1E840C1	E5C54040
09C1E3C5	40404040	C6C9C3C1	40E3C1E7
4C404040	40404040	404040F5	F3F5FAF
0F6F540C2	C5D5D5C9	C540D340	40404040
FC434040	40404040	58F4F2F1	48F4F640
40C4C4C0	40404040	4058F3F2	48F9F440
40404040	404040C8	40404040	40404040
40404040	40404040	404040C4	C5D0C1D9
03E84CF2	C9E9C540	40404040	F140C1D5
4043405F	F4F2F148	F4F64040	40404040

[illegible]

0053E0 F4F6C9D5 C2D3C540 C9C9C740 E5C1D540
 00540C F0FC4040 4C404040 40404040 40404040
 00542C 04040400 40404040 E0C04770 F05A47F0
 005440 D5404C40 4C404040 40404040 40404040
 005460 D6D5F3C8 C3E840D7 C1E84040 D5D64CD6
 005480 4040C6C5 C440E3C1 E7404040 F3C1E740
 0054AC 4C4040D5 C5E340D7 C1E84040 40404040
 0054CJ A0404040 40404040 40404040 40404040
 0054FC 404040C4 C5C7C1D9 E3D4C5D5 E34D5E06
 00550C 4C404040 F140C1D5 C440E3D6 E3C1C340
 005520 F4F6404C 40404040 40404040 40404040
 00554C 404040C4 C4C750E0 D0A458F0 1C2D9101
 00556J F7A447F0 F01A58E0 101C07FE D5D1F340
 005580 615C0CCC C0C00000 D0F0F34C 0A37D3D0
 0055AJ C0C0CCC0 C0C00000 F024CA07 90CEFC60
 0055FC 9ACEF06C C2001028 1017C200 10171016
 0055EC C0C0CCC0 C0C00000 C0C0C000 C0C0C0C0
 0056CC C0C0C0C0 --SAMF--
 0056EC C0C0C0C0 C0C0C000 C0C00000 C0C0C0C0
 00570C C0C0C0CC C0C053D8 C0C054CC C0C05CE8
 00572C 0C000000 0042146C 4C404058 F4F2F148
 00574C 0C005C00 C0005C94 0C0C5C60 C0C05C94
 00576C C0C0C0C0 C0C0C038 C0C0C0D8 C0C05A04
 005780 00005C94 C0C05860 C0C05872 C0C05904
 0057A0 000059AC C0C05586 000059A4 C0C059D6
 0057CC C0C05C4E C0C05PEF C0C05C6C C0C05C94
 0057EO 40202120 F0202120 000C0000 C0C0C004
 00580C 5810C1C8 4F10C086 94F1000 4110C088
 00582C 0C0C5288 C1C24110 C08E580C D1C1E540
 00584C 0A075020 C1C05880 C1C0C7D0 6C0C0C26
 00586C 5820C020 D5C06C00 6025C782 5P10C0C4
 005880 4110C090 C70005FC 5C00F3C8 45C0F00C
 0058A0 91801000 4780F016 41101004 412010E0
 0058C0 C70005F0 5C00F0C8 45C0F00C C0C0C0C0
 0058FC C71F1841 41F0C024 D2021041 F00158F0
 00590C C02A07FF 58C0D20C 5C00C1F8 5800C02C
 00592C C20C02CC 6C004025 5810C0C8 07F15820
 00594C 58CCC21C 5C00D1FC 5800C034 5C00C210
 00596C C03807F1 5820C038 D501A0C7 7018C782
 005980 5810C0C0 C7F15900 D20C5000 D20C5800
 0059AJ C01007F1 58C0D204 5C00C210 F271D1D0
 0059C0 D1D0A018 F176D1D0 D1D0F811 6017D1D6
 0059FC C06558F0 C04C07AF F671C1E0 601D5176
 005AC0 F271D1D0 7022F910 D1D0C06A 58F0C050
 005A2C 6017D1D6 5810C0C4 07F1F271 D1D07022
 005A4J 6021F176 C1D0D1D0 F811A017 D1D05810
 005A6C F8116C17 C1D0F271 D1D07022 F410C1D6
 005A80 D1D06017 F176D1D0 D1D0F176 D1D0D1D0
 005AAC FC41D1D2 6C19F175 C1D0D1D0 F8336000
 005AC0 D1D07024 F933D1D0 D1D0F811 6011D1C0
 005AEO F873C1D4 C1D0D209 D1D0C06E 4110C1E6
 0058D0 D1E2D2C8 6C837000 0211A0C1 7009F275
 0058P0 C1E0D1D0 C6109258 1C0C0C2C7 6C09D1E2
 005840 L1D0D201 6C7D1E2 C209C1E0 C0A4E110

4C4040F0 F3F0F2E2 E2C75CF4 F0F4F9F0
 4C404040 40404040 40404040 40404040
 F1404040 40404040 40404040 4040E2E2
 C5C1D4C5 40404040 40404040 404040D4
 C440C4C5 D7E240D7 C1E840C1 E5C54040
 C5C1E3C5 40404040 C6C9C3C1 40E3C1E7
 4C404040 40000000 47F0F04C 0A320000
 4C404040 40404040 40404040 40404040
 E3C1D34C C6C1D4C9 D3E840E2 C9E9C540
 C5C5F34C D7C1E840 40404058 F4F2F148
 4C404040 40404040 40404040 40404040
 1C044780 F04C9140 10024710 F04658E0
 F0C04770 F05A47F0 F04658E0 F06407FE
 3C000C48 07000000 00000000 000C5800
 58E01C18 48C0102E 06C01870 44C0F06C
 C0000C48 00000000 00000000 00000000
 C0C00000 00000000 00000000 00000000

C0C00000 00000000 00005000 0C00C000
 C0C05288 00005348 00000000 00000C4F
 F4F6C0C0 00000000 00000000 000058EE
 C0C05860 00005C60 00005C94 000000E8
 C0C058E8 000058EE 00005C60 00005C8E
 C0C0592E 0000591A 00005964 00005956
 C0C05A6A 00005A00 00005A2A 00005A54
 CF5C9C01 3C1C4020 58202021 2C482020
 5858C2D6 C7C5D540 5858C2C3 D3D6E2C5
 5800C1F8 184005FC 5C00F3D8 4500F00C
 C7C0C5F0 5000F008 4500F00C 00005348
 5800D208 5000D1F4 5800C01C 5000D208
 C7F15800 D1F4500C C2085800 D1C81840
 C0C0C0C0 0A025810 D1C84810 C08605F0
 C2D0F1D0 20005800 D1C81840 4110C090
 CAC02040E 5810D1C8 58F0C024 91201010
 101C45E0 F008502C D18C5870 D18C58F0
 5C00D20C 5810C00C 07F15800 D1F85000
 C0C0C540 60070772 50006038 60070772
 5810C010 07F1580C D1FC5030 D21C5810
 58C0C20C 5000D200 5800C03C 5000D20C
 D2105000 C2045800 C0405000 D2105810
 7C22F910 D1D0C06A 58F0C044 077FF871
 5810C048 07F1F271 D1D07022 F910D1D6
 C1D0C1D0 F811A017 D1D05810 C04807F1
 C7AFF871 D1D0601F F176D1D0 D1D0F811
 F511C1D6 C0A858F0 C05407AF F871D1D0
 C048C7F1 F871D1D0 6023F176 D1D0D1D0
 C06DFA11 6015D1D6 F275D170 7024FC51
 F8336039 D1D0490F 6039F275 D1D07024
 C1D04FA73 D1D06000 F433D1D4 6C09F275
 54C6C011 F873D1D8 6011FD51 D1D0A6U15
 F6C9D150 D1D0C0F10 92581030 D20760EF
 F1D97024 D209D1E0 C06E4110 D1E60F09
 F271C1E8 7022D0P03 D1E0C078 C03D1E0
 C1E6C0F0 D1E06009 06109258 1000D707

4WINKLE RIP VAN 0302SSG*40490
 00
0..0 1 SS
 N NAME M
 ONTHLY PAY NO O F DEPS PAY AVE
 FED TAX TAX RATE FICA TAX
 NET PAY00.....

DEPARTMENT TO TAL FAMILY SIZE
 1 AND TOTAL NET PAY \$421.

46
0.....
 0..00.....N..0-0..00.....
 /*.....00.....
0.....0-0.....
 ..0-K.....K.....

Q.....EY
 \$471. 46.....
 ..E.....*.....**.....*.....Y
-Q.....M ..\$Y.....*.....*
*.....
0
 ..*.....*.....*
0\$80PEN \$80CLOSE
 ..JH..... ..JH.....0E.....0
J..... ..0E.....0.....
 ..E.....J.....K.....-K.....J4.....E.....K
N.....-1.....J4E.....K.....JH
0E.....0JH.....0
0K.....J.....
0E.....0JH.....0
0.....K.....00.....E.....J.....J.....0
K.....E.....J8..... E.....K.....1.....J8E
 K.....K.....-N.....-
 ..K.....E.....J.....E.....K1.....J.....E.....K
1.....N.....-K.....E.....K.....E.....K
1.....K.....E.....KK.....E.....E.....K
1.....K.....E.....K9.....JQ.....0.....8
 J.....1.....J.....J.....-12.....J.....9.....JQ
0.....8.....J.....-J.....J.....-JQ.....1
 2.....J.....9.....JQ.....0.....E8.....J.....-J.....J.....8
 -JQ.....12.....J..... 9.....JQ.....0.....8.....J
 -1.....J.....J.....-18.....J.....-1.....J.....J
 8.....-JQ2.....J.....JQ8.....-JQ2.....J.....JQ
 JK.....-1.....J.....J.....J..... 8.....-JH.....-2.....J
 ..JK.....-1.....J.....J.....-JH.....J.....-JH.....-2.....J
 JQ.....J.....JH8.....-J-8.....JQ.....-J.....
 R.....JQ.....J.....JWJ.....J.....J.....K.....-
 JSK.....-K.....-A.....2JQ.....K.....J.....JW
 J.....J.....J.....K.....-RJS2.....JQ.....K.....J.....JW
 J.....K.....-XJSK.....J..... JW.....J.....-8.....K

005R6C	60F9D1E2	FA71D1D8	6017C7D5	D1C8C1C8	940F01DE	D203D1E0	C07C0E03	D1E0D1DE	-9JS8.JQ-P.JQJQ	..J.K.J.....J.J.
005R8C	C2016106	F1E2D709	D1F0C06E	4110D1E6	DF09C1E0	A000D41D	925R1000	D20761D3F	K./..JSK.J.....JW	..J.-.....S.K./.
005E8A	D1F2D209	D1E0C06E	4110C1E6	CF09D1E0	6011C610	925R1000	D207611A	D1E2FA33	JSK.J.....JW..J.S.K./..JS..
005PCC	6007F011	FA116CC1	6015C284	8CC060A8	5810D1CC	92F09000	58F01010	45E0F00C	-.....-K.....0.....0.....
005BEC	5020D1C0	58A0D1C0	5810D208	07F1D203	D1E0C078	DE03D1E0	6001D202	6172D1E1	E.J.....K.....K.J.-K./..J.
005CCJ	FA73D1D8	60C3D7C3	D1D8C1C8	940F01D0	C209C1E0	C36E4110	91E6D0F9	D1E0D10C	R.JQ-P.JQJQ..J.	K.J.....JW..J.J.
005C2Q	06105258	1000D207	41PAC1E2	D78380C0	413C9740	82845810	D1CC9760	800058F0K./..JSK..	/.. ..J.-.....0
005C4C	10104550	F00C5020	D1C05880	C1C0D201	6001C680	D2036003	C0825810	D20C07F10.E.J.....J.K.K.-.....K..1
005C6C	D7C16CC7	7C18D27E	8CC060B8	92408C7F	C2048080	807F5810	D1CC97F1	800058F0	K.-.....K.....	K.....J.-.....0
005CRQ	10104550	F00C5020	D1C05880	D1CC5810	D21CC7F1	0A0E500C	50085050	D00458200.E.J.....J.	K..1..E.E.E.E.....
005CAC	CCCC95CC	2CC00779	92FF2000	5610C0A8	50E0CC54	05F0912C	004847E0	F01658000.....0.....0.....0.....
005CCJ	P044582C	PC5058E0	D054C7FE	9620C0A8	416CC004	4110C004	4170C0A8	06700550f.....f.....	-.....f.....f.....
005CE0	58401000	1E485040	1CCC8716	5C004180	D18C4170	D1CF051C	58308000	1F0P5000f.....f.....	J.....J.....f.....
005C00	A0008784	1C00D208	D2C8C05C	5E60C1C4	5870C18C	5880D1C0	58E0DC54	07FEF01AK.K.*.-JD	..J.....J.....0..
005D20	0A32C00C	47F0FC84	0A320000	47F0F01A	C9D1C4C6	C107C9E9	F3F99130	1002471000.....00..	1JDFAPIZ39.....
005C40	F024C0C7	5C0E00EC	58E01C18	06ECC700	1C17E000	18CC18DC	43C01017	41E00010	D.....0.....K..0.....0.....
005D60	43DEF0F7	19C4C780	F05046E0	F0400A32	43DEF107	43C01016	19C04770	F0769180	..07.....0E..0..	..1.....0.....
005D8C	10164780	FC769208	1C28CACC	91801002	4710F076	0A0742C0	10284200	101604000.....0.....0.....0.....
005FAC	51801002	4710FC8A	0AC7C201	1C261C03	92C11028	58E01028	D2021029	101950E00.....K.....K.....E..
005DCC	1018C6EC	58CCF0EC	44001022	58E0FCF4	0A0007FE	91801002	4710FC8E	0AC791010.....0040.....0.....
005DEC	10154780	PCD949FE	10195102	1026C78E	47FCFC0A	91011077	078E1200	4780F0E40M.....	..00.....00.....
005E00	18F007FF	92881028	0A0007FE	00005768	C00C5548	R0005C86	F2C2C1F9	F8F7F6F5	..0.....*..28A98765
005E2C	F4F3C3F1	4E60F040	92D8C3CB	C38883A8	A99E388	00131308	47F0F014	47F0F252	43C1+-0 ..L.C...	..T.....00..02..
005E40	47F0F027	47F0FC14	47F0F014	0A32C9D1	C7C6C9C5	EAE9F3F9	0009903E	F26C9180	..00..00..00..1J	GFIEWZ39....2...
005E60	1C49471C	FC4A58A0	102C50A0	10584AA0	104A50A0	10609143	10274710	F1CC47F00.....f.....	..E.-.....1..0
005E80	F0A0988C	1C58868C	FC8E90B0	1C589110	1C154780	F07E18D0	41A00100	18CA47C0	0.....0.E.....0.....0.....
005EAO	F076D2FF	DC008000	1ADA868A	F06441CC	00FF44C0	F25A98AE	F26C4400	105407FE	0.K.....0.....2...2.....
005ECO	91081C49	4780F098	94F71049	47F0F1CC	91C81064	4710F0A2	0A009180	100247100.....7...01.0.....0.....
005EEC	F0AC0AC7	91011004	4780F098	47F0F236	91C81003	4780F0C4	91801064	4780F000	0.....0.....0..02.0D.....0.....
005FC0	58E01C44	47F0F0F4	91921003	4780F110	918C1064	4780F14C	58E01064	98AD26C00U.....1..1.....2.....
005F2Q	90F1F100	58101C80	C700C5FE	99F1E00C	9CADF26C	47F0F154	00000000	00000000	..11.....1.....	..2...01.....
005F4C	CCCC00CC	CCCC000C	91431005	4780F158	49810000	4980F260	4780F13C	48P0105E0.....1.....2...1.....
005F60	4980F260	4720F124	4740F13C	56401C49	47FCF158	91041064	4780F098	58E01050	..2...1.. 1..01.....0Q...E
005F80	47F0F0E4	91201064	4780F158	96101049	58B01C80	418A00C0	50801058	91401049	..00U.....1.....f.....f.....
005FA0	4710F178	4A80104A	50801060	47F0F18A	48C01C4A	48C10000	1AC850C0	10609740	..1.....E.-..01.A.....E.-..
005FC0	104948E0	103489E0	000843E0	F2634180	CC0518CD	43C91038	43081043	19C047702.....2.....
005FEO	F18642E8	1C3888E0	0CC846E0	F19C41CC	CC0142C8	1038D503	1C3C1036	4700F202	1.....2.....1.....N.....2.....
006000	96081049	9001F280	98AD26C	4100F284	411CF264	0A028801	F2809120	101047102...2...2..	..?.....7.....
006020	F07E9140	10274780	F1FA94FF	102747F0	F0A09108	10494710	F21A9108	10644780	0.....1.....0.....	0.....2.....0.....
006040	F21A5880	1C80D202	1C811020	5080102C	CA0C9110	10494780	F22A9710	104947F0	2.....K.....E.....2.....0.....
006060	F0889680	1C499880	105847F0	F0569180	10264710	F1CC9120	10274710	F1CC94FE	0.....00.....1.....1.....
00608C	1C49964C	1C2747F0	F1CC2203	10581060	C7FED200	D0008000	00000711	5858C20601..K.....	..K.....*8RO
0060AD	C7C3C540	CC050000	CC0C50C0	CC0C5768	CC035548	800058F6	0C000C00	00000000	PEN ..E.....E.....6.....
0060CC	FFCCG000	00000000	00000000	00CC0000	CC0C0000	00000C03	FFC4E2C1	C5F0F3F11L.DSAE031
0060E0	05F047F0	F018C9D3	C2C4E2C1	C5F0F3F1	F4F3F0F3	61F0F761	F7F41800	47F0F022	..0.00..1L8DSAE031	4303/07/74...00..
006100	41000001	C5F050E0	F0F09005	F0F24144	CC0G9110	4014478C	FC09108	401547100E.0...02..0.....0.....
00612C	F03C912C	4C024780	F02C4100	001047F0	F0305811	00044154	00004850	F11A5855	0... ..0.....0	0.....f1...
006140	C0004155	0C001255	4780F0AC	1835D501	FCE43C00	4780F05A	41330032	47F0F0480.....N..	0U.....0.....00..
006160	5050F0EA	C5033006	F0F04780	F09E4403	CC064403	000A1852	92F35000	D2065001	EEO.N.....0.....0E.K.E..
006180	500C12C0	4780F0C9A	490CF0E6	4780F0C92	94C15C02	47F0F0C9E	94015001	47F0F09E	E.....0.....0M..0.	..E..00...E..00..
0061A0	96015000	58F0F0EE	9805F0F2	58F0F0EA	C7FF9900	00184840	F11C1634	4110F10A	..f.....0...02..00.1.....1.....
0061CC	0A0291CE	FC484780	FC0C5810	F11259F0	F116C701	1CC21002	4111003C	50D1C0000.....1..0	1.P.....EJ..
0061F0	C7FF1900	CA060000	000005F0	000183E0	C0384680	F19C41CC	000142C8	103895030.....1.....N..
0062CC	5858C2C3	DAC2C5D9	0CC0C000	CCCC0000	5858C2C3	D6C7C5D9	00000000	00000000	*8COBER.....	*8COBER.....

COBOL SYNTAX DEBUG

Practical Exercise Booklet

INSTRUCTIONS TO THE STUDENT:

This booklet contains three (3) COBOL Source program listings, each with an Associated Compiler Error Diagnostic Listing.

You are to correct the syntax errors by;

1. Circle the portion of the source statement in error,
2. pencil in the correction to be made on the source listing.

When you correct all the errors in the first practical exercise turn to the back of this booklet and check your answers against the school solution. If you have any questions concerning this exercise see your instructor for assistance.

Follow this procedure for all three practical exercises in this booklet.

IA-01-03-03

1

```

00001 IDENTIFICATION DIVISION.
00002 PROGRAM-ID. COBSYN1.
00003 AUTHOR. SP6 DW GROSS, SOFTWARE.
00004 INSTALLATION. CSD, USAIA, FT HARRISON, IN 46216.
00005 DATE-WRITTEN. MAR 3 1978.
00006 DATE-COMPILED. JUN 15, 1978
00007 SECURITY. UNCLASSIFIED.
00008 *REMARKS. THIS PROGRAM LISTS ALL ORDERS RECEIVED
00009 * EACH DAY. IT LISTS THE PART NUMBER, CUSTOMER
00010 * NAME, AND CUSTOMER ADDRESS.
00011 *
00012 * WS-ORDER-EOF-SW IS USED IN THE PROCEDURE
00013 * DIVISION TO SIGNIFY WHEN THE END OF FILE
00014 * CONDITION HAS BEEN REACHED.
00015 ENVIRONMENT DIVISION.
00016 CONFIGURATION SECTION.
00017 SOURCE-COMPUTER. IBM-360-H40.
00018 OBJECT-COMPUTER. IBM-360-H40.
00019 SPECIAL-NAMES.
00020 CO1 IS TOP-OF-PAGE.
00021 INPUT-OUTPUT SECTION.
00022 FILE-CONTROL.
00023 SELECT IM-ORDER-FILE
00024 ASSIGN TO UT-S-SYS006.
00025 SELECT OR-ORDERS-LISTING
00026 ASSIGN TO UT-S-SYS005.
00027 DATA DIVISION.
00028 FILE SECTION.
00029 FD IM-ORDER-FILE,
00030 BLOCK CONTAINS 1 RECORDS,
00031 RECORD CONTAINS 80 CHARACTERS,
00032 LABEL RECORD IS STANDARD.
00033 DATA RECORD IS IM01-ORDER-RCD.
00034 01 IM01-ORDER-RCD.
00035 05 IM01-PART-NO PIC 9(4).
00036 05 IM01-CUSTOMER-NAME PIC X(18).
00037 05 IM01-CUSTOMER-ADDRESS PIC X(18).
00038 05 IM01-UNIT-OF-ISSUE PIC XX.
00039 05 IM01-QTY-ORDERED PIC 999.
00040 05 IM01-PRICE-ITEM PIC 9(3)V99.
00041 05 IM01-CUSTOMER-CODE PIC X.
00042 05 FILLER PIC X(31).
00043 FD OR-ORDERS-LISTING,
00044 BLOCK CONTAINS 1 RECORDS,
00045 RECORD CONTAINS 133 CHARACTERS,
00046 LABEL RECORD IS STANDARD,
00047 DATA RECORD IS OR01-ORDER-LINE.
00048 01 OR01-ORDER-LINE PIC X(133).
00049 WORKING-STORAGE SECTION.
00050 77 WS-ORDER-EOF-SW PIC XXX
00051 VALUE 'OFF'.
00052 01 WS-HEADING-LINE.
00053 05 FILLER PIC X(33)
00054 VALUE SPACES.

```



```

00055      05 FILLER                                PIC X(7)
00056      VALUE 'PART NO'.
00057      05 FILLER                                PIC X(11)
00058      VALUE SPACES.
00059      05 FILLER                                PIC X(13)
00060      VALUE 'CUSTOMER NAME'.
00061      05 FILLER                                PIC X(12)
00062      VALUE SPACES.
00063      05 FILLER                                PIC X(16)
00064      VALUE 'CUSTOMER ADDRESS'.
00065      05 FILLER                                PIC X(41)
00066      VALUE SPACES.
00067  01 WS-DETAIL-LINE.
00068      05 FILLER                                PIC X(35)
00069      VALUE SPACES.
00070      05 WS-DL-PART-NO                          PIC 9(4).
00071      05 FILLER                                PIC 1(X0)
00072      VALUE SPACES.
00073      05 WS-DL-CUSTOMER-NAME                    PIC X(18).
00074      05 FILLER                                PIC X(8)
00075      VALUE SPACES.
00076      05 WS-DL-CUSTOMER-ADDRESS                 PIC X(18).
00077      05 FILLER                                PIC X(40)
00078      VALUE SPACES
00079  PROCEDURE DIVISION.
00080  0010-DRIVER.
00081      OPEN INPUT IM-ORDER-FILE
00082      OUTPUT OR-ORDERS-LISTING.
00083      WRITE OR01-ORDER-LINE FROM WS-HEADING-LINE
00084      AFTER ADVANCING TOP-OF-PAGE.
00085      PERFORM 0020-READ-LIST-RTN THRU 0020-EIXT
00086      UNTIL WS-ORDER-EOF-SW EQUAL 'ON'.
00087      CLOSE IM-ORDER-FILE
00088      OR-ORDERS-LISTING.
00089  0010-EXIT.
00090      STOP RUN.
00091  0020-READ-LIST-RTN.
00092      READ IM-ORDER-FILE
00093      AT END
00094      MOVE 'ON' TO WS-ORDER-EOF-SW
00095      GO TO 0020-EXIT.
00096      MOVE IM01-PART-NO TO WS-DL-PART-NU.
00097      MOVE IM01-CUSTOMER-NAME TO WS-DL-CUSTOMER-NAME.
00098      MOVE IM01-CUSTOMER-ADDRESS TO WS-DL-CUSTOMER-ADDRESS.
00099      WRITE OR01-ORDER-LINE FROM WS-DETAIL-LINE
00100      AFTER ADVANCING 2 LINE.
00101  0020-EXIT.
00102      EXIT.

```

CARD ERROR MESSAGE

33	IKF10041-E	INVALID WORD DATA . SKIPPING TO NEXT RECOGNIZABLE WORD.
23	IKF21461-W	RECORD SIZE IN RECORD-CONTAINS CLAUSE DISAGREES WITH COMPUTED RECORD SIZE. USING MAXIMUM COMPUTED SIZE.
71	IKF20391-C	PICTURE CONFIGURATION ILLEGAL. PICTURE CHANGED TO 9 UNLESS USAGE IS 'DISPLAY-ST', THEN L1618DZ98DZ9.
71	IKF21291-C	VALUE CLAUSE LITERAL DOES NOT CONFORM TO PICTURE. CHANGED TO ZERO.
79	IKF10431-W	END OF SENTENCE SHOULD PRECEDE PROCEDURE . ASSUMED PRESENT.
85	IKF30011-E	0020-EIXY NOT DEFINED. THRU OPTION DISCARDED.
96	IKF30011-E	WS-DL-PART-NU NOT DEFINED. DISCARDED.
99	IKF40031-E	EXPECTING NEW STATEMENT. FOUND LINES . DELETING TILL NEXT VERB OR PROCEDURE-NAME.

1

```

00001 PROGRAM-ID. COBSYN2.
00002 IDENTIFICATION DIVISION.
00003 AUTHOR. SP6 DW GROSS.
00004 INSTALLATION. CSD, USAIA, FT HARRISON, IN 46216.
00005 DATE WRITTEN. MAR 3, 1978.
00006 DATE-COMPILED. JUN 15, 1978
00007 SECURITY. UNCLASSIFIED.
00008 *REMARKS. THIS PROGRAM LISTS ALL PERSONNEL IN THE
00009 * BATTALION WITH OVER 75 MONTHS SERVICE.
00010 *
00011 ENVIRONMENT DIVISION.
00012 CONFIGURATION SECTION.
00013 SOURCE-COMPUTER. IBM-360-H40.
00014 OBJECT-COMPUTER. IBM-360-H40.
00015 SPECIAL-NAMES.
00016 CO1 IS TOP-OF-PAGE.
00017 INPUT-OUTPUT SECTION.
00018 FILE-CONTROL.
00019 SELECT IM-PERSONNEL-MASTER-FILE
00020 ASSIGN TO TU-S-SYS006.
00021 SELECT OR-PRINTER
00022 ASSIGN TO UT-S-SYS005.
00023 DATA DIVISION.
00024 FILE SECTION.
00025 FD IM-PERSONNEL-MASTER-FILE
00026 BLOCK CONTAINS 1 RECORDS,
00027 RECORD CONTAINS 80 CHARACTERS,
00028 LABEL RECORD IS STANDARD,
00029 DATA RECORD IS IM01-PERSONNEL-MASTER-RCD.
00030 01 IM01-PERSONNEL-MASTER-RCD.
00031 05 IM01-BATTALION PIC XX.
00032 05 IM01-COMPANY PIC X.
00033 05 IM01-SSAN PIC 9(9).
00034 05 IM01-NAME PIC X(18).
00035 05 IM01-PAY-GRADE PIC XY.
00036 05 IM01-RANK PIC X(3).
00037 05 IM01-ETS PIC X(6).
00038 05 IM01-DATE-OF-ENTRY PIC X(6).
00039 05 IM01-MONTHS-OF-SERVICE PIC 9(3).
00040 05 IM01-NUMBER-OF-DEPENDENTS PIC 99.
00041 05 IM01-PREVIOUS-ASSIGN-INFO PIC X(28).
00042 FD OR-PRINTER,
00043 BLOCK CONTAINS 1 RECORDS,
00044 RECORD CONTAINS 133 CHARACTERS,
00045 LABEL RECORD IS STANDARD,
00046 DATA RECORD IS OR01-PRINTER-RCD.
00047 01 OR01-PRINTER-RCD PIC X(13X).
00048 WORKING-STORAGE SECTION.
00049 77 WS-MASTER-FILE-EOF-SWITCH PIC XXX
00050 VALUE 'OFF'
00051 01 WS-HEADING-LINE.
00052 05 FILLER PIC X(53)
00053 VALUE SPACES.
00054 05 FILLER PIC X(5)

```

00055		VALUE 'OVER 75 MONTHS OF SERVICE'.	
00056	05 FILLER		PIC X(55)
00057		VALUE SPACES	
00058	01 WS-DETAIL-LINE.		
00059	05 FILLER		PIC X(36)
00060		VALUE SPACES.	
00061	05 WS-DL-NAME		PIC X(18).
00062	05 FILLER		PIC X(10)
00063		VALUE SPACES.	
00064	05 WS-DL-RANK		PIC XXX.
00065	05 FILLER		PIC X(9)
00066		VALUE SPACES.	
00067	05 WS-DL-SSAN		PIC 9(9).
00068	05 FILLER		PIC X(7)
00069		VALUE SPACES.	
00070	05 WS-DL-MONTHS-OF-SERVICE		PIC ZZ9.
00071	05 FILLER		PIC X(38).
00072		VALUE SPACES	
00073	PROCEDURE DIVISION.		
00074	0010-DRIVER.		
00075	OPEN INPUT IM-PERSONNEL-MASTER-FILE		
00076	OUTPUT OR-PRINTER.		
00077	WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE		
00078	AFTER ADVANCING TOP-OF-PAGE.		
00079	PERFORM 0020-READ-LIST-RTN THRU 0020-EXIT		
00080	UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.		
00081	CLOSE IM-PERSONNEL-MASTER-FILE.		
00082	OR-PRINTER.		
00083	0010-EXIT.		
00084	STOP RUN.		
00085	0020-READ-LIST-RTN.		
00086	READ IM-PERSONNEL-MASTER-FILE		
00087	AT END		
00088	MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH		
00089	GO TO 0020-EXIT.		
00090	IF IM01-MONTHS-OF-SERVICE GREATER THAN 75		
00091	THEN		
00092	MOVE IM01-NAME TO WS-DL-NAME		
00093	MOVE IM01-RANK TO WS-DL-RANK		
00094	MOVE IM01-SSAN TO WS-DL-SSAN		
00095	MOVE WS-DL-MONTHS-OF-SERVICE TO		
00096	IM01-MONTHS-OF-SERVICE		
00097	WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE		
00098	AFTER ADVANCING 2 LINES.		
00099	0020-EXIT.		
00100	EXIT.		

CARD ERROR MESSAGE

1	IKF11291-C	ID DIV. HEADER EXTRANEIOUS, MISSING OR MISPLACED. ONE ASSUMED PRESENT.
5	IKF10871-W	' DATE ' SHOULD NOT BEGIN A-MARGIN.
20	IKF11551-W	DEVICE CLASS INVALID IN SYSTEM-NAME. SKIPPING TO NEXT FIELD.
35	IKF20391-C	PICTURE CONFIGURATION ILLEGAL. PICTURE CHANGED TO 9 UNLESS USAGE IS 'DISPLAY-ST', THEN L1618DZ98DZ9.
19	IKF21461-W	RECORD SIZE IN RECORD-CONTAINS CLAUSE DISAGREES WITH COMPUTED RECORD SIZE. USING MAXIMUM COMPUTED SIZE.
47	IKF20391-C	PICTURE CONFIGURATION ILLEGAL. PICTURE CHANGED TO 9 UNLESS USAGE IS 'DISPLAY-ST', THEN L1618DZ98DZ9.
51	IKF10431-W	END OF SENTENCE SHOULD PRECEDE 01 . ASSUMED PRESENT.
21	IKF21461-W	RECORD SIZE IN RECORD-CONTAINS CLAUSE DISAGREES WITH COMPUTED RECORD SIZE. USING MAXIMUM COMPUTED SIZE.
54	IKF21261-C	VALUE CLAUSE LITERAL TOO LONG. TRUNCATED TO PICTURE SIZE.
58	IKF10431-W	END OF SENTENCE SHOULD PRECEDE 01 . ASSUMED PRESENT.
72	IKF10041-E	INVALID WORD VALUE . SKIPPING TO NEXT RECOGNIZABLE WORD.
79	IKF10071-W	SLASH NOT PRECEDED BY A SPACE. ASSUME SPACE.
79	IKF30011-E	002 NOT DEFINED. STATEMENT DISCARDED.
79	IKF10071-W	MINUS SIGN NOT PRECEDED BY A SPACE. ASSUME SPACE.
79	IKF30011-E	READ-LIST-RTN NOT DEFINED.
82	IKF10861-W	OR-PRINTER SHOULD BEGIN A-MARGIN.
95	IKF40521-E	DNM=1-358 (EDI MAY NOT BE TARGET FIELD FOR DNM=2-283 (NE) IN MOVE STATEMENT, AND IS DISCARDED.

329

IA-01-03-03

1

```

00001 IDENTIFICATION DIVISION.
00002 PROGRAM ID. COBSYN3.
00003 AUTHOR. SP6 DW GROSS.
00004 INSTALLATION. CSD, USAIA, FT HARRISON, IN 46216.
00005 DATE-WRITTEN. MAR 3, 1978.
00006 DATE-COMPILED. JUN 15, 1978
00007 SECURITY. UNCLASSIFIED.
00008 *REMARKS. THIS PROGRAM PRODUCES THE DEPENDENT INFORMATION
00009 * REPORT. IT LISTS ALL PERSONNEL IN THE BATTALION
00010 * WITH 5 OR MORE DEPENDENTS BY NAME, RANK, COMPANY,
00011 * AND SIZE OF FAMILY.
00012 *
00013 * WS-MASTER-FILE-EOF-SWITCH IS USED TO SIGNIFY
00014 * WHEN THE END OF FILE CONDITION HAS BEEN
00015 * ENCOUNTERED.
00016 * WS-TOTAL-DEPENDENTS IS USED TO ACCUMULATE THE
00017 * DEPENDENTS FOR THE PURPOSE OF PRINTING IT IN
00018 * THE TOTAL LINE.
00019 *
00020 ENVIRONMENT DIVISION.
00021 CONFIGURATION SECTION.
00022 SOURCE COMPUTER. IBM-360-H40.
00023 OBJECT-COMPUTER. IBM-360-H40.
00024 SPECIAL-NAMES.
00025 CO1 IS TOP-OF-PAGE.
00026 INPUT-OUTPUT SECTION.
00027 FILE CONTROL.
00028 SELECT IM-PERSONNEL-MASTER-FILE
00029 ASSIGN TO UT-S-SYS006.
00030 SELECT OR-PRINTER
00031 ASSIGN TO UT-S-SYS005.
00032 DATA DIVISION.
00033 FILE SECTION.
00034 FD IM-PERSONNEL-MASTER-FILE,
00035 BLOCK CONTAINS 1 RECORDS,
00036 RECORD CONTAINS 80 CHARACTERS,
00037 LABEL RECORD IS STANDARD,
00038 DATA RECORD IS IM01-PERSONNEL-MASTER-RCO.
00039 01 IM01-PERSONNEL-MASTER-RCO
00040 05 IM01-BATTALION PIC XX.
00041 05 IM01-COMPANY PIC X.
00042 05 IM01-SSAN PIC 9(9).
00043 05 IM01-NAME PIC X(18).
00044 05 IM01-PAY-GRADE PIC XX.
00045 05 IM01-RANK PIC XXX.
00046 05 IM01-ETS PIC X(6).
00047 05 IM01-DATE-OF-ENTRY PIC X(6).
00048 05 IM01-MONTHS-OF-SERVICE PIC 999.
00049 05 IM01-NUMBER-OF-DEPENDENTS PIC XX.
00050 05 IM01-PREVIOUS-ASSIGN-INFO PIC X(28).
00051 FD OR-PRINTER,
00052 BLOCK CONTAINS 1 RECORDS,
00053 RECORD CONTAINS 133 CHARACTERS,
00054 LABEL RECORD IS STANDARD,

```

7

IA-01-C3-C3

00055	DATA RECORD IS DRO1-PRINTER-RCD.		
00056	01 DRO1-PRINTER-RCD		PIC X(133).
00057	WORKING-STORAGE SECTION.		
00058	77 WS-LINE-COUNTER		PIC Z9
00059		VALUE ZEROS.	
00060	77 WS-TOTAL-DEPENDENTS		PIC ZZ9
00061		VALUE ZEROS.	
00062	77 WS-MASTER-FILE-EOF-SWITCH		PIC XXX
00063		VALUE 'OFF'.	
00064	01 WS-HEADING-LINE-1.		
00065	05 FILLER		PIC X(53)
00066		VALUE SPACES.	
00067	05 FILLER		PIC X(28)
00068		VALUE 'DEPENDENT INFORMATION REPORT'.	
00069	05 FILLER		PIC X(52)
00070		VALUE SPACES.	
00071	01 WS-HEADING-LINE-2.		
00072	05 FILLER		PIC X(38)
00073		VALUE SPACES.	
00074	05 FILLER		PIC X(4).
00075		VALUE 'NAME'.	
00076	05 FILLER		PIC X(18)
00077		VALUE SPACES.	
00078	05 FILLER		PIC X(4)
00079		VALUE 'RANK'.	
00080	05 FILLER		PIC X(10)
00081		VALUE SPACES.	
00082	05 FILLER		PIC X(7)
00083		VALUE 'COMPANY'.	
00084	05 FILLER		PIC X(7)
00085		VALUE SPACES.	
00086	05 FILLER		PIC X(11)
00087		VALUE 'FAMILY SIZE'.	
00088	05 FILLER		PIC X(34)
00089		VALUE SPACES.	
00090	01 WS-DETAIL-LINE.		
00091	05 FILLER		PIC X(32)
00092		VALUE SPACES.	
00093	05 WS-DL-NAMEC		PIC X(18)
00094	05 FILLER		PIC X(10)
00095		VALUE SPACES.	
00096	05 WS-DL-RANK		PIC XXX.
00097	05 FILLER		PIC X(14)
00098		VALUE SPACES.	
00099	05 WS-DL-COMPANY		PIC X.
00100	05 FILLER		PIC X(15)
00101		VALUE SPACE..	
00102	05 WS-DL-FAMILY-SIZE		PIC ZZ9.
00103	05 FILLER		PIC X(36)
00104		VALUE SPACES.	
00105	01 WS-TOTAL-LINE.		
00106	05 FILLER		PIC X(32)
00107		VALUE SPACES.	
00108	05 FILLER		PIC X(34)
00109		VALUE 'TOTAL NUMBER OF DEPENDENTS WITHIN'.	
00110	05 FILLER		PIC X(18)
00111		VALUE 'THE BATTALION IS'.	

00112 05 WS-TL-NUMBER-OF-DEPENDENTS PIC ZZ9.
00113 05 FILLER PIC X(46)
00114 VALUE SPACES.
00115 PROCEDURE DIVISION.
00116 0010-DRIVER.
00117 OPEN INPUT IM-PERSONNEL-MASTER-FILE
00118 OUTPUT OR-PRINTER.
00119 PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
00120 PERFORM 0020-READ-WRITE-RTN THRU 0020-EXIT
00121 UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
00122 MOVE WS-TOTAL-DEPENDENTS TO WS-TL-NUMBER-OF-DEPENDENTS.
00123 WRITE OR01-PRINTER-RCD FROM WS-TOTAL-LINE
00124 AFTER ADVANCING 3 LINES.
00125 CLOSE IM-PERSONNEL-MASTER-FILE,
00126 OR-PRINTER.
00127 0010-EXIT.
00128 EXIT.
00129 0020-READ-WRITE-RTN.
00130 READ IM-PERSONNEL-MASTER-FILE
00131 AT END
00132 MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
00133 GO TO 0020-EXIT.
00134 ADD IM01-NUMBER-OF-DEPENDENTS TO WS-TOTAL-DEPENDENTS.
00135 IF IM01-NUMBER-OF-DEPENDENTS LESS THAN 4
00136 THEN
00137 GO TO 0020-EXIT.
00138 IF WS-LINE-COUNTER GREATER THAN 15
00139 THEN
00140 PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
00141 MOVE IM01-NAME TO WS-DL-NAME.
00142 MOVE IM01-RANK TO WS-DL-RANK.
00143 MOVE IM01-COMPANY TO WS-DL-COMPANY.
00144 ADD 1, IM01-NUMBER-OF-DEPENDENTS GIVING WS-DL-FAMILY-SIZE.
00145 WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
00146 AFTER ADVANCING 2 LINES.
00147 ADD 1 TO WS-LINE-COUNTER.
00148 0020-EXIT.
00149 EXIT.
00150 0030-HEADING-RTN.
00151 MOVE 0 TO WS-LINE-COUNTER.
00152 WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-1
00153 AFTER ADVANCING TOP-OF-PAGE.
00154 WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-2
00155 AFTER ADVANCING 3 LINES.
00156 0030-EXIT.
00157 EXIT.

CARD	ERROR MESSAGE
2	IKF1087I-W
2	IKF1097I-E
22	IKF1087I-W
22	IKF1004I-E
27	IKF1087I-W
27	IKF1004I-E
28	IKF1003I-W
40	IKF1043I-W
75	IKF1004I-E
94	IKF1043I-W
101	IKF1081I-W
101	IKF1080I-W
122	IKF3001I-E
122	IKF3001I-E
134	IKF4019I-E
134	IKF4019I-E
141	IKF3001I-E
144	IKF4019I-E
144	IKF5011I-W
147	IKF4019I-E

* PROGRAM * SHOULD NOT BEGIN A-MARGIN.
 PROGRAM-ID MISSING OR MISPLACED. IF PROGRAM-ID DOES NOT IMMEDIATELY FOLLOW
 IDENTIFICATION DIVISION, IT WILL BE IGNORED.
 * SOURCE * SHOULD NOT BEGIN A-MARGIN.
 INVALID WORD SOURCE . SKIPPING TO NEXT RECOGNIZABLE WORD.
 * FILE * SHOULD NOT BEGIN A-MARGIN.
 INVALID WORD FILE . SKIPPING TO NEXT RECOGNIZABLE WORD.
 FILE-CONTROL PARAGRAPH NAME MISSING. ASSUMED PRESENT.
 END OF SENTENCE SHOULD PRECEDE 05 . ASSUMED PRESENT.
 INVALID WORD VALUE . SKIPPING TO NEXT RECOGNIZABLE WORD.
 END OF SENTENCE SHOULD PRECEDE 05 . ASSUMED PRESENT.
 PERIOD NOT FOLLOWED BY SPACE. ASSUME END OF SENTENCE.
 PERIOD PRECEDED BY SPACE. ASSUME END OF SENTENCE.
 WS-TOTAL-DEPENDENTS NOT DEFINED. DISCARDED.
 WS-TL-NUMBER-OF-DEPENDENTS NOT DEFINED.
 DNM=1-427 (AN) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT.
 ARBITRARILY SUBSTITUTING TALLY .
 DNM=2-81 (NE) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT. ARBITRARILY
 SUBSTITUTING TALLY .
 WS-OL-NAME NOT DEFINED. DISCARDED.
 DNM=1-427 (AN) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT.
 ARBITRARILY SUBSTITUTING TALLY .
 AN INTERMEDIATE RESULT OR A SENDING FIELD MIGHT HAVE ITS HIGH ORDER DIGIT POSITION
 TRUNCATEL.
 DNM=2-49 (NE) MAY NOT BE USED AS ARITHMETIC OPERAND IN ADD STATEMENT. ARBITRARILY
 SUBSTITUTING TALLY .

333

10

JA-01-03-03

1

```

00001 IDENTIFICATION DIVISION.
00002 PROGRAM-ID. COBSYN1.
00003 AUTHOR. SP6 DW GROSS, SOFTWARE.
00004 INSTALLATION. CSD, USAIA, FT HARRISON, IN 46216.
00005 DATE-WRITTEN. MAR 3 1978.
00006 DATE-COMPILED. JUN 21, 1978
00007 SECURITY. UNCLASSIFIED.
00008 *REMARKS. THIS PROGRAM LISTS ALL ORDERS RECEIVED
00009 * EACH DAY. IT LISTS THE PART NUMBER, CUSTOMER
00010 * NAME, AND CUSTOMER ADDRESS.
00011 *
00012 * WS-ORDER-EOF-SW IS USED IN THE PROCEDURE
00013 * DIVISION TO SIGNIFY WHEN THE END OF FILE
00014 * CONDITION HAS BEEN REACHED.
00015 ENVIRONMENT DIVISION.
00016 CONFIGURATION SECTION.
00017 SOURCE-COMPUTER. IBM-360-H40.
00018 OBJECT-COMPUTER. IBM-360-H40.
00019 SPECIAL-NAMES.
00020 COI IS TOP-OF-PAGE.
00021 INPUT-OUTPUT SECTION.
00022 FILE-CONTROL.
00023 SELECT IM-ORDER-FILE.
00024 ASSIGN TO UT-S-SYS006.
00025 SELECT OR-ORDERS-LISTING
00026 ASSIGN TO UT-S-SYS005.
00027 DATA DIVISION.
00028 FILE SECTION.
00029 FD IM-ORDER-FILE,
00030 BLOCK CONTAINS 1 RECORDS,
00031 RECORD CONTAINS 80 CHARACTERS,
00032 LABEL RECORD IS STANDARD,
00033 DATA RECORD IS IM01-ORDER-RCD.
00034 01 IM01-ORDER-RCD.
00035 05 IM01-PART-NO PIC 9(4).
00036 05 IM01-CUSTOMER-NAME PIC X(18).
00037 05 IM01-CUSTOMER-ADDRESS PIC X(18).
00038 05 IM01-UNIT-OF-ISSUE PIC XX.
00039 05 IM01-QTY-ORDERED PIC 999.
00040 05 IM01-PRICE-ITEM PIC 9(3)V99.
00041 05 IM01-CUSTOMER-CODE PIC X.
00042 05 FILLER PIC X(31).(29)
00043 FD OR-ORDERS-LISTING,
00044 BLOCK CONTAINS 1 RECORDS,
00045 RECORD CONTAINS 133 CHARACTERS,
00046 LABEL RECORD IS STANDARD,
00047 DATA RECORD IS OR01-ORDER-LINE.
00048 01 OR01-ORDER-LINE PIC X(133).
00049 WORKING-STORAGE SECTION.
00050 77 WS-ORDER-EOF-SW PIC XXX
00051 VALUE 'OFF'.
00052 01 WS-HEADING-LINE.
00053 05 FILLER PIC X(33)
00054 VALUE SPACES.

```

```

00055      05 FILLER                                PIC X(7)
00056      VALUE 'PART NO'.
00057      05 FILLER                                PIC X(11)
00058      VALUE SPACES.
00059      05 FILLER                                PIC X(13)
00060      VALUE 'CUSTOMER NAME'.
00061      05 FILLER                                PIC X(12)
00062      VALUE SPACES.
00063      05 FILLER                                PIC X(16)
00064      VALUE 'CUSTOMER ADDRESS'.
00065      05 FILLER                                PIC X(41)
00066      VALUE SPACES.
00067 01 WS-DETAIL-LINE.
00068      05 FILLER                                PIC X(35)
00069      VALUE SPACES.
00070      05 WS-DL-PART-NO                          PIC 9(4).
00071      05 FILLER                                PIC (11X0) X(10)
00072      VALUE SPACES.
00073      05 WS-DL-CUSTOMER-NAME                    PIC X(18).
00074      05 FILLER                                PIC X(8)
00075      VALUE SPACES.
00076      05 WS-DL-CUSTOMER-ADDRESS                PIC X(18).
00077      05 FILLER                                PIC X(40)
00078      VALUE SPACES.
00079 PROCEDURE DIVISION.
00080 0010-DRIVER.
00081     OPEN INPUT IM-ORDER-FILE
00082     OUTPUT OR-ORDERS-LISTING.
00083     WRITE OR01-ORDER-LINE FROM WS-HEADING-LINE
00084     AFTER ADVANCING TOP-OF-PAGE.
00085     PERFORM 0020-READ-LIST-RTN THRU 0020-EXIT EXIT
00086     UNTIL WS-ORDER-EOF-SW EQUAL 'ON'.
00087     CLOSE IM-ORDER-FILE
00088     OR-ORDERS-LISTING.
00089 0010-EXIT.
00090     STOP RUN.
00091 0020-READ-LIST-RTN.
00092     READ IM-ORDER-FILE
00093     AT END
00094         MOVE 'ON' TO WS-ORDER-EOF-SW
00095         GO TO 0020-EXIT.
00096     MOVE IM01-PART-NO TO WS-DL-PART-NO NO
00097     MOVE IM01-CUSTOMER-NAME TO WS-DL-CUSTOMER-NAME.
00098     MOVE IM01-CUSTOMER-ADDRESS TO WS-DL-CUSTOMER-ADDRESS.
00099     WRITE OR01-ORDER-LINE FROM WS-DETAIL-LINE
00100     AFTER ADVANCING 2 LINES LINES.
00101 0020-EXIT.
00102     EXIT.

```

335



1

```

00001  PROGRAM-ID.  COBSYN2.  IDENTIFICATION DIVISION.
00002  IDENTIFICATION DIVISION.  PROGRAM-ID.  COBSYN2.
00003  AUTHOR.  SP6 DW GROSS.
00004  INSTALLATION.  CSD, USAIA, FT HARRISON, IN 46216.
00005  DATE WRITTEN.  MAR 3, 1978. DATE-WRITTEN.
00006  DATE-COMPILED.  JUN 21, 1978
00007  SECURITY.  UNCLASSIFIED.
00008  *REMARKS.  THIS PROGRAM LISTS ALL PERSONNEL IN THE
00009  *  BATTALION WITH OVER 75 MONTHS SERVICE.
00010  *
00011  ENVIRONMENT DIVISION.
00012  CONFIGURATION SECTION.
00013  SOURCE-COMPUTER.  IBM-360-H40.
00014  OBJECT-COMPUTER.  IBM-360-H40.
00015  SPECIAL-NAMES.
00016  CO1 IS TOP-OF-PAGE.
00017  INPUT-OUTPUT SECTION.
00018  FILE-CONTROL.
00019  SELECT IM-PERSONNEL-MASTER-FILE
00020  ASSIGN TO (U-S-SYS006)  UT-S-SYS006.
00021  SELECT OR-PRINTER
00022  ASSIGN TO UT-S-SYS005.
00023  DATA DIVISION.
00024  FILE SECTION.
00025  FD  IM-PERSONNEL-MASTER-FILE
00026  BLOCK CONTAINS 1 RECORDS,
00027  RECORD CONTAINS 80 CHARACTERS,
00028  LABEL RECORD IS STANDARD,
00029  DATA RECORD IS IM01-PERSONNEL-MASTER-RCO.
00030  01  IM01-PERSONNEL-MASTER-RCO.
00031  05  IM01-BATTALION  PIC XX.
00032  05  IM01-COMPANY  PIC X.
00033  05  IM01-SSAN  PIC 9(9).
00034  05  IM01-NAME  PIC X(18).
00035  05  IM01-PAY-GRADE  PIC (XY) XX.
00036  05  IM01-RANK  PIC X(3).
00037  05  IM01-ETS  PIC X(6).
00038  05  IM01-DATE-OF-ENTRY  PIC X(6).
00039  05  IM01-MONTHS-OF-SERVICE  PIC 9(3).
00040  05  IM01-NUMBER-OF-DEPENDENTS  PIC 99.
00041  05  IM01-PREVIOUS-ASSIGN-INFO  PIC X(28).
00042  FD  OR-PRINTER,
00043  BLOCK CONTAINS 1 RECORDS,
00044  RECORD CONTAINS 133 CHARACTERS,
00045  LABEL RECORD IS STANDARD,
00046  DATA RECORD IS OR01-PRINTER-RCO.
00047  01  OR01-PRINTER-RCO  PIC (X(13X))  X(133).
00048  WORKING-STORAGE SECTION.
00049  77  WS-MASTER-FILE-EOF-SWITCH  PIC XXX
00050  VALUE 'OFF'  VALUE 'OFF'.
00051  01  WS-HEADING-LINE.
00052  05  FILLER  PIC X(53)
00053  VALUE SPACES.
00054  05  FILLER  PIC (X(5))  X(25)

```

13

IA-01-03-03

336

```

00055          VALUE 'OVER 75 MONTHS OF SERVICE'.
00056          PIC X(55)
00057          VALUE SPACES
00058 01 WS-DETAIL-LINE.
00059          05 FILLER          PIC X(36)
00060          VALUE SPACES.
00061          05 WS-DL-NAME      PIC X(18).
00062          05 FILLER          PIC X(10)
00063          VALUE SPACES.
00064          05 WS-DL-RANK      PIC XXX.
00065          05 FILLER          PIC X(9)
00066          VALUE SPACES.
00067          05 WS-DL-SSAN      PIC 9(9).
00068          05 FILLER          PIC X(7)
00069          VALUE SPACES.
00070          05 WS-DL-MONTHS-OF-SERVICE PIC 229.
00071          05 FILLER          PIC X(38).
00072          VALUE SPACES
00073 PROCEDURE DIVISION.
00074 0010-DRIVER.
00075     OPEN INPUT IM-PERSONNEL-MASTER-FILE
00076           OUTPUT OR-PRINTER.
00077     WRITE CRO1-PRINTER-RCD FROM WS-HEADING-LINE
00078           AFTER ADVANCING TOP-OF-PAGE.
00079     PERFORM 0020-READ-LIST-RTN THRU 0020-EXIT 0020-READ-LIST-RTN
00080           UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
00081     CLOSE IM-PERSONNEL-MASTER-FILE CLOSE IM-PERSONNEL-MASTER-FILE
00082           OR-PRINTER OR-PRINTER.
00083 0010-EXIT.
00084     STOP RUN.
00085 0020-READ-LIST-RTN.
00086     READ IM-PERSONNEL-MASTER-FILE
00087           AT END
00088           MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
00089           GO TO 0020-EXIT.
00090     IF IM01-MONTHS-OF-SERVICE GREATER THAN 75
00091       THEN
00092         MOVE IM01-NAME TO WS-DL-NAME
00093         MOVE IM01-RANK TO WS-DL-RANK
00094         MOVE IM01-SSAN TO WS-DL-SSAN
00095         MOVE WS-DL-MONTHS-OF-SERVICE TO IM01-MONTHS-OF-SERVICE MOVE IM01-MONTHS-OF-SERVICE TO
00096           IM01-MONTHS-OF-SERVICE WS-DL-MONTHS-OF-SERVICE
00097         WRITE CRO1-PRINTER-RCD FROM WS-DETAIL-LINE
00098           AFTER ADVANCING 2 LINES.
00099 0020-EXIT.
00100     EXIT.

```

1

```

00001 IDENTIFICATION DIVISION. PROGRAM-ID.
00002 PROGRAM-ID. COBSYN3.
00003 AUTHOR. SP6 DW GROSS.
00004 INSTALLATION. CSD, USAIA, FT HARRISON, IN 46216.
00005 DATE-WRITTEN. MAR 3, 1978.
00006 DATE-COMPILED. JUN 21, 1978
00007 SECURITY. UNCLASSIFIED.
00008 *REMARKS. THIS PROGRAM PRODUCES THE DEPENDENT INFORMATION
00009 * REPORT, IT LISTS ALL PERSONNEL IN THE BATTALION
00010 * WITH 5 OR MORE DEPENDENTS BY NAME, RANK, COMPANY,
00011 * AND SIZE OF FAMILY.
00012 *
00013 * WS-MASTER-FILE-EOF-SWITCH IS USED TO SIGNIFY
00014 * WHEN THE END OF FILE CONDITION HAS BEEN
00015 * ENCOUNTERED.
00016 * WS-TOTAL-DEPENDENTS IS USED TO ACCUMULATE THE
00017 * DEPENDENTS FOR THE PURPOSE OF PRINTING IT IN
00018 * THE TOTAL LINE.
00019 *
00020 ENVIRONMENT DIVISION.
00021 CONFIGURATION SECTION.
00022 SOURCE COMPUTER. IBM-360-H40. SOURCE-COMPUTER.
00023 OBJECT-COMPUTER. IBM-360-H40.
00024 SPECIAL-NAMES.
00025 CO1 IS TOP-OF-PAGE.
00026 INPUT-OUTPUT SECTION.
00027 FILE CONTROL.
00028 SELECT IM-PERSONNEL-MASTER-FILE
00029 ASSIGN TO UT-S-SYS006.
00030 SELECT OR-PRINTER
00031 ASSIGN TO UT-S-SYS005.
00032 DATA DIVISION.
00033 FIVE SECTION. FILE-SECTION.
00034 FD IM-PERSONNEL-MASTER-FILE,
00035 BLOCK CONTAINS 1 RECORDS,
00036 RECORD CONTAINS 80 CHARACTERS,
00037 LABEL RECORD IS STANDARD,
00038 DATA RECORD IS IM01-PERSONNEL-MASTER-RCO.
00039 01 IM01-PERSONNEL-MASTER-RCO IM01-PERSONNEL-MASTER-RCO.
00040 05 IM01-BATTALION PIC XX.
00041 05 IM01-COMPANY PIC X.
00042 05 IM01-SSAN PIC 9(9).
00043 05 IM01-NAME PIC X(18).
00044 05 IM01-PAY-GRADE PIC XX.
00045 05 IM01-RANK PIC XXX.
00046 05 IM01-ETS PIC X(6).
00047 05 IM01-DATE-OF-ENTRY PIC X(6).
00048 05 IM01-MONTHS-OF-SERVICE PIC 999.
00049 05 IM01-NUMBER-OF-DEPENDENTS PIC XX. 99.
00050 IM01-PREVIOUS-ASSIGN-INFO PIC X(29).
00051 FD OR-PRINTER,
00052 BLOCK CONTAINS 1 RECORDS,
00053 RECORD CONTAINS 133 CHARACTERS,
00054 LABEL RECORD IS STANDARD,

```

00055	DATA RECORD IS 0001-PRINTER-RCD.		
00056	01	0001-PRINTER-RCD	PIC X(133).
00057	WORKING-STORAGE SECTION.		
00058	77	WS-LINE-COUNTER	PIC (29) 99
00059		VALUE ZEROS.	
00060	77	WS-TOTAL-DEPENDENTS	PIC (229) 999
00061		VALUE ZEROS.	
00062	77	WS-MASTER-FILE-EOF-SWITCH	PIC XXX
00063		VALUE 'OFF'.	
00064	01	WS-HEADING-LINE-1.	
00065	05	FILLER	PIC X(53)
00066		VALUE SPACES.	
00067	05	FILLER	PIC X(28)
00068		VALUE 'DEPENDENT INFORMATION REPORT'.	
00069	05	FILLER	PIC X(52)
00070		VALUE SPACES.	
00071	01	WS-HEADING-LINE-2.	
00072	05	FILLER	PIC X(38)
00073		VALUE SPACES.	
00074	05	FILLER	PIC X(41) PIC X(4)
00075		VALUE 'NAME'.	
00076	05	FILLER	PIC X(18)
00077		VALUE SPACES.	
00078	05	FILLER	PIC X(4)
00079		VALUE 'RANK'.	
00080	05	FILLER	PIC X(10)
00081		VALUE SPACES.	
00082	05	FILLER	PIC X(7)
00083		VALUE 'COMPANY'.	
00084	05	FILLER	PIC X(7)
00085		VALUE SPACES.	
00086	05	FILLER	PIC X(11)
00087		VALUE 'FAMILY SIZE'.	
00088	05	FILLER	PIC X(34)
00089		VALUE SPACES.	
00090	01	WS-DETAIL-LINE.	
00091	05	FILLER	PIC X(32)
00092		VALUE SPACES.	
00093	05	WS-DL-NAMEC	PIC X(18) PIC X(18) WS-DL-NAME
00094	05	FILLER	PIC X(10)
00095		VALUE SPACES.	
00096	05	WS-DL-RANK	PIC XXX.
00097	05	FILLER	PIC X(14)
00098		VALUE SPACES.	
00099	05	WS-DL-COMPANY	PIC X.
00100	05	FILLER	PIC X(15)
00101		VALUE SPACE.	SPACES.
00102	05	WS-DL-FAMILY-SIZE	PIC 2229.
00103	05	FILLER	PIC X(36)
00104		VALUE SPACES.	
00105	01	WS-TOTAL-LINE.	
00106	05	FILLER	PIC X(32)
00107		VALUE SPACES.	
00108	05	FILLER	PIC X(34)
00109		VALUE 'TOTAL NUMBER OF DEPENDENTS WITHIN'.	
00110	05	FILLER	PIC X(18)
00111		VALUE 'THE BATTALION IS'.	

00112 05 WS-TL-NUMBER-OF-DEPENDENTS PIC ZZZ. WS-TL-NUMBER OF ..
 00113 05 FILLER PIC X(48)
 00114 VALUE SPACES.
 00115 PROCEDURE DIVISION.
 00116 0010-DRIVER.
 00117 OPEN INPUT IM-PERSONNEL-MASTER-FILE
 00118 OUTPUT OR-PRINTER.
 00119 PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
 00120 PERFORM 0020-READ-WRITE-RTN THRU 0020-EXIT
 00121 UNTIL WS-MASTER-FILE-EOF-SWITCH EQUAL 'ON'.
 00122 MOVE WS-TOTAL-DEPENDENTS TO WS-TL-NUMBER-OF-DEPENDENTS. MOVE WS-TOTAL
 00123 WRITE OR01-PRINTER-RCD FROM WS-TOTAL-LINE
 00124 AFTER ADVANCING 3 LINES.
 00125 CLOSE IM-PERSONNEL-MASTER-FILE,
 00126 OR-PRINTER.
 00127 0010-EXIT.
 00128 EXIT.
 00129 0020-READ-WRITE-RTN.
 00130 REAC IM-PERSONNEL-MASTER-FILE
 00131 AT END
 00132 MOVE 'ON' TO WS-MASTER-FILE-EOF-SWITCH
 00133 GO TO 0020-EXIT.
 00134 ADD IM01-NUMBER-OF-DEPENDENTS TO WS-TOTAL-DEPENDENTS.
 00135 IF IM01-NUMBER-OF-DEPENDENTS LESS THAN 4
 00136 THEN
 00137 GO TO 0020-EXIT.
 00138 IF WS-LINE-COUNTER GREATER THAN 15
 00139 THEN
 00140 PERFORM 0030-HEADING-RTN THRU 0030-EXIT.
 00141 MOVE IM01-NAME TO WS-DL-NAME.
 00142 MOVE IM01-RANK TO WS-DL-RANK.
 00143 MOVE IM01-COMPANY TO WS-DL-COMPANY.
 00144 ADD 1, IM01-NUMBER-OF-DEPENDENTS GIVING WS-DL-FAMILY-SIZE.
 00145 WRITE OR01-PRINTER-RCD FROM WS-DETAIL-LINE
 00146 AFTER ADVANCING 2 LINES.
 00147 ADD 1 TO WS-LINE-COUNTER.
 00148 0020-EXIT.
 00149 EXIT.
 00150 0030-HEADING-RTN.
 00151 MOVE 0 TO WS-LINE-COUNTER.
 00152 WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-1
 00153 AFTER ADVANCING TOP-OF-PAGE.
 00154 WRITE OR01-PRINTER-RCD FROM WS-HEADING-LINE-2
 00155 AFTER ADVANCING 3 LINES.
 00156 0030-EXIT.
 00157 EXIT.


CHECK PROBLEM

No. _____

SCHOOLCRAFT MICHIGAN _____ 19 _____ 74-827
724

PAY TO THE ORDER OF _____ \$ _____

_____ DOLLARS

 Kalamazoo County State Bank
SCHOOLCRAFT • MICHIGAN.

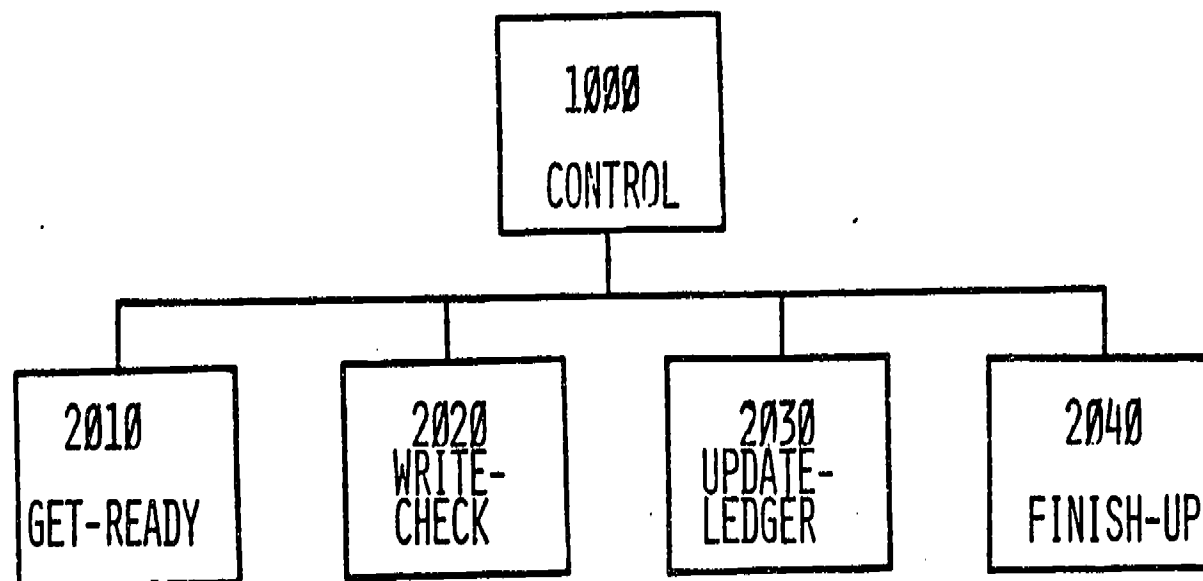
⑈0724⑈0827⑈

[illegible]

INDIANAPOLIS POWER & LIGHT COMPANY					KEEP THIS PART		
P. O. BOX 15938 INDIANAPOLIS, INDIANA 46206							
SERV.	RATE	SERVICE DATES		METER READINGS		KWH USED	AMOUNT
		FROM	TO	PREVIOUS	PRESENT		
ELPS	809	908	5568	211	643	2391	
FA			FUEL CHARGE			345	
TX			INDIANA SALES TAX			109	
BILLING DATE						5-08-7	NET DUE
							2345
LATE PAYMENT CHARGE ADDED AFTER						5-28-7	GROSS DUE
							2948

AUGUST, 197						
S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

CHECK PROBLEM HIERARCHY CHART



PROGRAM NAME: CHECK PROBLEMSTUDENT NAME: KUCINICHDATE: JAN 79

IPO

MODULE NUMBER: 2021MODULE NAME: WRITE-CHECKPAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. OPENED CHECK-BOOK	ADD 1 TO PREVIOUS-CK-NO.	1. WRITTEN-CHECK
1.1 BLANK-CHECK	MOVE PREVIOUS-CHK-NO TO CHECK-NO.	1.1 CHECK-NO
1.2 CHECK-LEDGER	MOVE BILL-PAYEE TO CHECK-PAYEE.	1.2 CHECK-DATE
1.2.1 PREVIOUS- CHK-NO	MOVE BILL-AMOUNT TO CHECK-AMT-NUMERIC.	1.3 CHECK-PAYEE
2. TODAYS-DATE	MOVE BILL-AMOUNT TO CHECK-AMT-ALPHA.	1.4 CHECK-AMOUNT
	MOVE TODAYS-DATE TO CHECK-DATE	
3. OPENED UNPAID-BILL	SIGN CHECK.	1.4.1 CHK-AMT- NUMERIC
3.1 BILL-PAYEE	EXIT.	1.4.2 CHK-AMT-ALPHA
3.2 BILL-AMOUNT		1.5 SIGNATURE



PROGRAM NAME: CHECK PROBLEM

STUDENT NAME: KUCINICH

DATE: JAN 79

IPO

MODULE NUMBER: 2030

MODULE NAME: UPDATE-LEDGER

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CHECK	MOVE CHECK-NO TO LEDGER-CHK-NO.	1. UPDATED-LEDGER
1.1 CHECK-NO	MOVE CHECK-DATE TO LEDGER-DATE.	1.1 LEDGER-CHK-NO
1.2 CHECK-DATE	MOVE CHECK-PAYEE TO LEDGER-PAYEE.	1.2 LEDGER-DATE
1.3 CHECK-PAYEE	MOVE CHECK-AMT-NUMERIC TO LEDGER-AMT-	1.3 LEDGER-PAYEE
	OF-CHK.	1.4 LEDGER-AMT-OF-CHK
1.4 CHECK-AMT-	SUBTRACT CHK-AMT-NUMERIC FROM LEDGER-OLD-	1.5 LEDGER-NEW-BALANCE
NUMERIC	BALANCE.	
2. CHECK-LEDGER	MOVE LEDGER-OLD-BALANCE TO LEDGER-	
	NEW-BALANCE.	
2.1 LEDGER-OLD-BALANCE	EXIT.	

PROGRAM NAME: CHECK PROBLEM

STUDENT NAME: KUCINICH

DATE: JAN 79

IPO

MODULE NUMBER: 27119

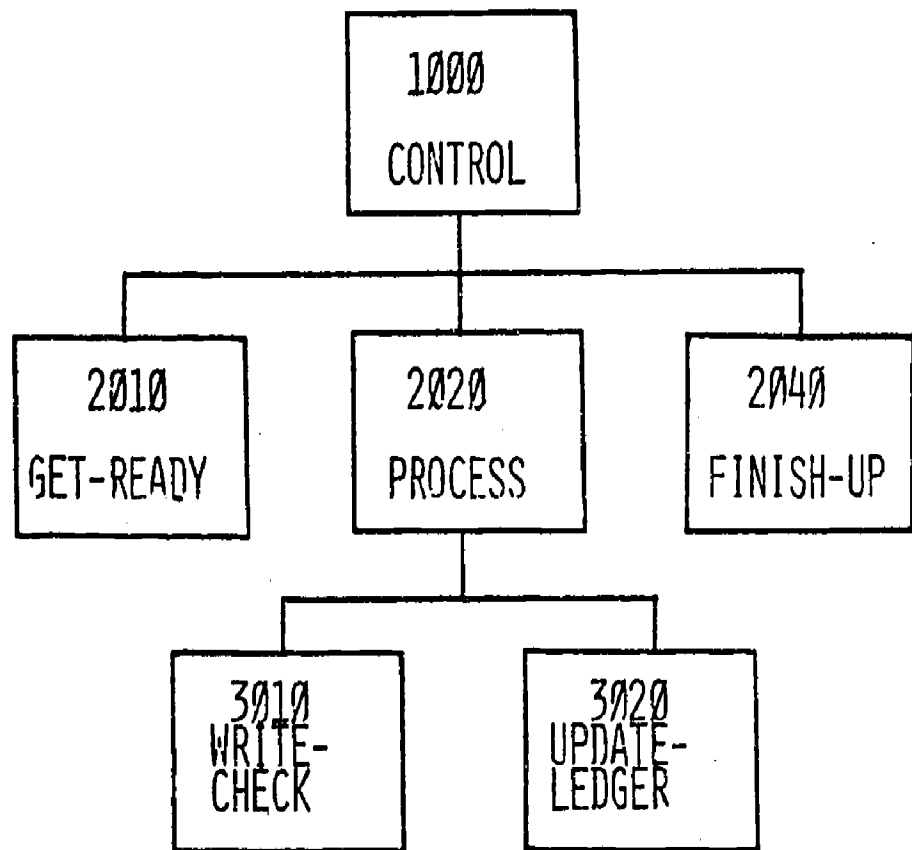
MODULE NAME: FINISH-UP

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. OPENED CHECK-BOOK	REMOVE WRITTEN CHECK FROM CHECK BOOK.	1. WRITTEN-CHECK
2. OPENED UNPAID-BILL	MARK BILL PAID.	2. CLOSED-CHECK-BOOK
	CLOSED CHECK-BOOK.	2.1 UPDATED-LEDGER
	EXIT.	3. PAID-BILL



CHECK PROBLEM MODIFICATION HIERARCHY CHART



IPO

MODULE NUMBER: 1090

STUDENT NAME: KUCINICH

MODULE NAME: CONTROL

DATE: JAN 79

PAGE 1 of 1

[illegible]

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

IA-01-01-24

2

PROGRAM NAME: CHECK PROBLEM MODIFICATION

STUDENT NAME: KUCINICH

DATE: JAN 79

IPO

MODULE NUMBER: 2020

MODULE NAME: PROCESS

PAGE 1 of 1

[illegible]

CHECK PROBLEM MODIFICATION

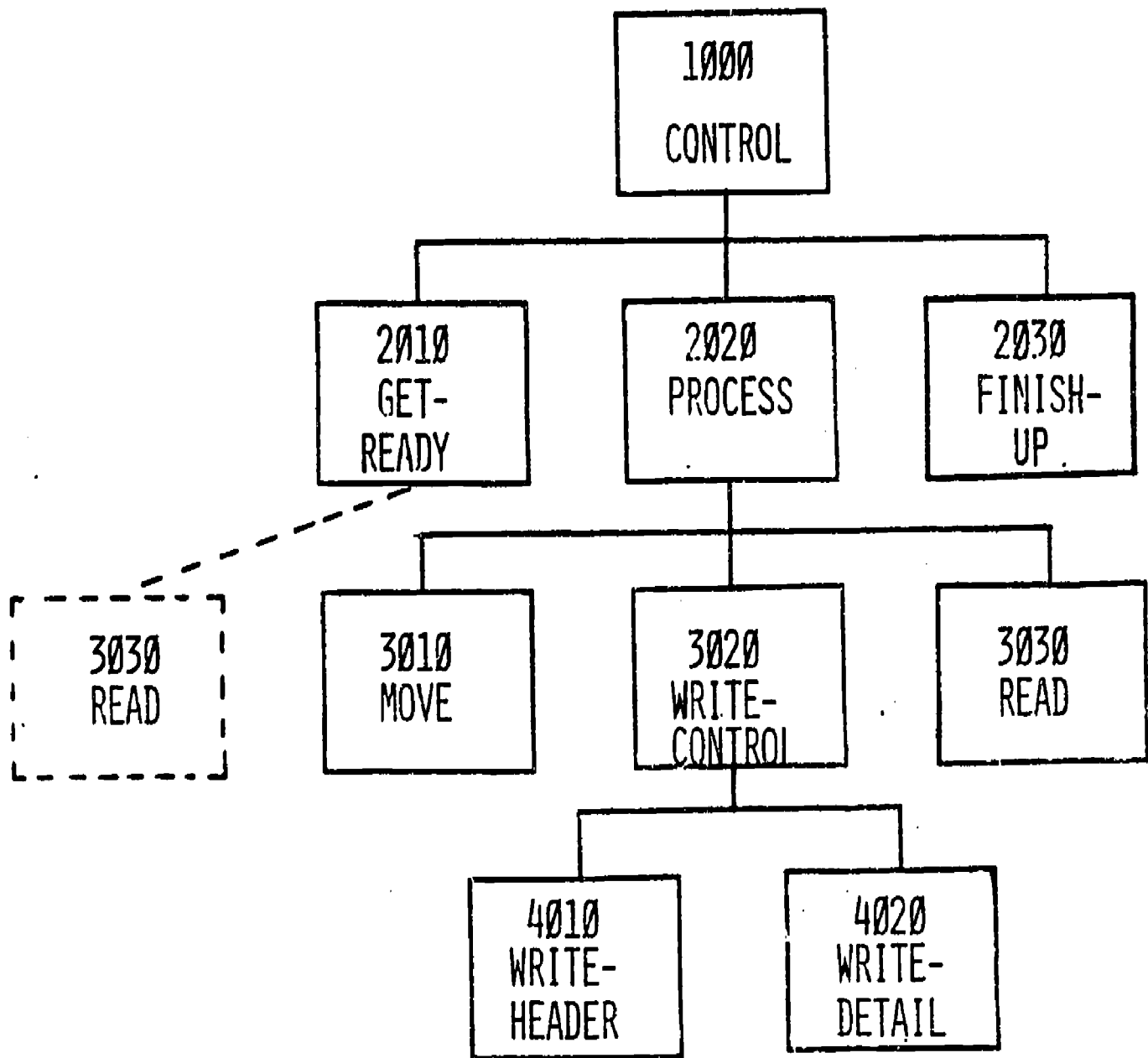
Now that you have solved the problem of writing a check and updating the check ledger to pay a bill, you will modify your solution to follow this process until you have payed all the bills.

You will assume that there is more than enough in the account to cover any and all bills that you must pay.

To accomplish this, you must redo your HIPO, and modify or create new IPOs.

1A-01-02-22

PROBLEM SOLVING PE #1 HIERARCHY CHART



DATE: FEB 79

IPO

MODULE NUMBER: 1999

MODULE NAME: CONTROL

PAGE 1 of 1

[illegible]

121-Ø13-14Ø1-Ø8Ø-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

IA-01-03-14

2

W
W
7

061

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2010

MODULE NAME: GET-READY

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL- MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE. OPEN OUTPUT PRINT-FILE.	1. OPENED PERSONNEL- MASTER-FILE
2. CLOSED PRINT-FILE	MOVE 'OFF' TO EOF-SW. MOVE 'OFF' TO HEADER-SW. PERFORM 3030-READ.	2. OPENED PRINT-FILE
EOF-SW	EXIT.	EOF-SW = 'OFF'
HEADER-SW		HEADER-SW = 'OFF'

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: EEB 79

IPO

MODULE NUMBER: 2020

MODULE NAME: PROCESS

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
	PERFORM 3010-MOVE.	
	PERFORM 3020-WRITE-CONTROL.	
	PERFORM 3030-READ.	
	EXIT.	



PROGRAM NAME: PROBLEM SOLVING PF #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2030

MODULE NAME: FINISH-UP

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. OPENED PERSONNEL- MASTER-FILE	CLOSE PERSONNEL-MASTER-FILE.	1. CLOSED PERSONNEL- MASTER-FILE
2. OPENED PRINT-FILE	CLOSE PRINT-FILE.	2. CLOSED PRINT-FILE
	EXIT.	

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 3929

MODULE NAME: WRITE-CONTROL

PAGE 1 of 1

342

INPUT	PROCESS	OUTPUT
	IF HEADER-SW = 'OFF'	
	THEN PERFORM 4010-WRITE-HEADER	
	MOVE 'ON' TO HEADER-SW	
	ELSE NEXT SENTENCE.	
	PERFORM 4020-WRITE-DETAIL.	
HEADER-SW = 'OFF' OR 'ON'	EXIT.	HEADER-SW = 'OFF' OR 'ON'

PROGRAM NAME: PROBLEM SOLVING DF #1STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 3030MODULE NAME: READPAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	READ PERSONNEL-MASTER-FILE	1. PERSONNEL-MASTER-
	AT END	RECORD
	MOVE 'ON' TO EOF-SW.	
	EXIT.	
EOF-SW = 'OFF' OR 'ON'		EOF-SW = 'OFF' OR 'ON'

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 4010

MODULE NAME: WRITE-HEADER

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
	WRITE HEADER-RECORD AFTER ADVANCING TO	1. HEADER-RECORD
	TOP OF PAGE.	
	EXIT.	
1. HEADER-RECORD		
1.1 'RANK'		
1.2 'NAME'		

344

PROGRAM NAME: PROBLEM SOLVING PE #1

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 4020

MODULE NAME: WRITE-DETAIL

PAGE 1 of 1

[illegible]

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY."

IA-01-03-14

10

PROBLEM SOLVING PE #2

- I. The records section here at Fort Harrison is not satisfied with the report from PE #1. They have asked for the following modifications.

Count the lines on each page and print only twenty (20) detail lines on each page.

Format the output report as follows;

FORT HARRISON PERSONNEL LISTING

RANK	NAME
XXX	XXXXXXXXXXXXXXXXXXXXX
XXX	XXXXXXXXXXXXXXXXXXXXX

Print the heading lines at the top of each new page.

1A-01-02-7

PROGRAM NAME: PROBLEM SOLVING PE #2

IPO

MODULE NUMBER: 2010

STUDENT NAME: KUCINICH

MODULE NAME: GET-READY

DATE: FEB 79

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL- MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE. OPEN OUTPUT PRINT-FILE.	1. OPENED PERSONNEL- MASTER-FILE
		2. OPENED PRINT-FILE
2. CLOSED PRINT-FILE	MOVE 'OFF' TO EOF-SW. MOVE 99 TO LINE-CNTR.	
EOF-SW	PERFORM 3030-READ.	EOF-SW = 'OFF'
LINE-CNTR	EXIT.	LINE-CNTR = 99

PROGRAM NAME: PROBLEM SOLVING PE #2

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 3020

MODULE NAME: WRITE-CONTROL

PAGE 1 of 1

348

INPUT	PROCESS	OUTPUT
	IF LINE-CNTR IS GREATER THAN 19	
	THEN PERFORM 4010-WRITE-HEADER	
	MOVE 0 TO LINE-CNTR	
	ELSE NEXT SENTENCE.	
	PERFORM 4020-WRITE-DETAIL.	
LINE-CNTR = (N)	ADD 1 TO LINE-CNTR.	LINE-CNTR = (N+1) OR 0
	EXIT.	



PROGRAM NAME: PROBLEM SOLVING PE #2

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 4019

MODULE NAME: WRITE-HEADER

PAGE 1 of 1

INPUT	PROCESS	OUTPUT
	WRITE HEADER-RECORD-1 AFTER ADVANCING TO	1. HEADER-RECORD-1
	TOP OF PAGE.	
	WRITE HEADER-RECORD AFTER ADVANCING 2	2. HEADER-RECORD
	LINES.	
	EXIT.	
1. HEADER-RECORD-1		
1.1 'FORT HARRISON		
PERSONNEL LISTING'		
2. HEADER-RECORD		
2.1 'RANK'		
2.2 'NAME'		

349

383

PROBLEM SOLVING PE #3

I INTRODUCTION.

The Battalion Headquarters at Fort Harrison has asked for a listing of all personnel assigned to or stationed here for accountability purposes. They would also like a count of all records listed.

II INPUT/OUTPUT.

INPUT: Input will consist of a personnel Master file, which contains records on all personnel stationed at or assigned to the Battalion.

Personnel Master Records;

1. SOCIAL SECURITY ACCOUNT NUMBER
2. NAME
3. BATTALION
4. RANK
5. NUMBER OF DEPENDENTS

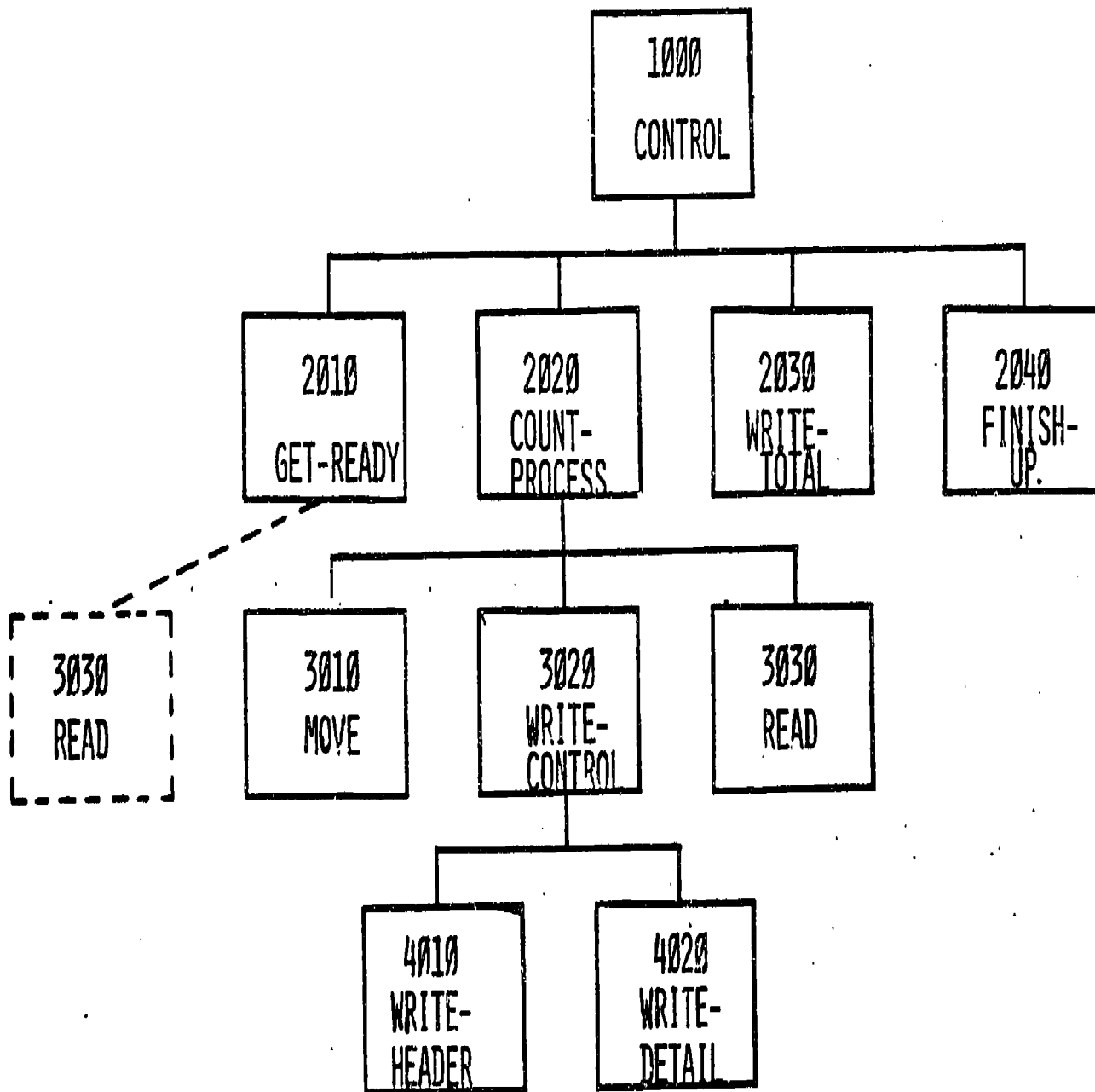
OUTPUT: Output will consist of a printed listing with a heading at the top of each page. Print a detail line for each record read, and a total line showing the number of records processed. Write 20 detail lines per page (double spaced) then start at the top of the next page with the header. The total line will be printed only once at the end of the report.

Format the output report as follows;

SSN	NAME	RANK	NR DEPS
999999999	xxxxxxxxxxxxxxxxxxxxx	xxx	99
999999999	xxxxxxxxxxxxxxxxxxxxx	xxx	99
999999999	xxxxxxxxxxxxxxxxxxxxx	xxx	99

TOTAL RECORDS PRINTED 999

PROBLEM SOLVING PE #3 HIERARCHY CHART



PROGRAM NAME: PROBLEM SOLVING PE #3MODULE NUMBER: 1000STUDENT NAME: KUCINICHMODULE NAME: CONTROLDATE: FEB 79PAGE 1 of 1

IPO

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	PERFORM 2010-GET-READY.	1. PRINTED-LISTING
FILE	PERFORM 2020-COUNT-PROCESS UNTIL	
	EOF-SW = 'ON'.	
	PERFORM 2030-WRITE-TOTAL.	
	PERFORM 2040-FINISH-UP.	
EOF-SW = 'OFF'	STOP RUN.	EOF-SW = 'ON'

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

IA-01-03-26

388

2 389

PROGRAM NAME: PROBLEM SOLVING PE #3STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 2010MODULE NAME: GET-READYPAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL- MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE.	1. OPENED PERSONNEL- MASTER-FILE
2. CLOSED PRINT-FILE	OPEN OUTPUT PRINT-FILE.	2. OPENED PRINT-FILE
	MOVE 99 TO LINE-CNTR.	
	MOVE 'OFF' TO EOF-SW.	
	MOVE 0 TO PERS-CNTR.	
LINE-CNTR	PERFORM 3030-READ.	LINE-CNTR = 99
EOF-SW	EXIT.	EOF-SW = 'OFF'
PERS-CNTR		PERS-CNTR = 0

PROGRAM NAME: PROBLEM SOLVING PE #3

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 2020

MODULE NAME: COUNT-PROCESS

PAGE 1 of 1

354

INPUT	PROCESS	OUTPUT
	ADD 1 TO PERS-CNTR.	
	PERFORM 3010-MOVE.	
	PERFORM 3020-WRITE-CONTROL.	
	PERFORM 3030-READ.	
PERS-CNTR = (N)	EXIT.	PERS-CNTR = (N+1)



PROGRAM NAME: PROBLEM SOLVING PE #3STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 2030MODULE NAME: WRITE-TOTALPAGE 1 of 1

INPUT	PROCESS	OUTPUT
	MOVE PERS-CNTR TO TOTAL-PERS.	1. TOTAL-LINE
	WRITE TOTAL-LINE AFTER ADVANCING 2 LINES.	
	EXIT.	
1. TOTAL-LINE		
1.1 'TOTAL RECORDS		
PRINTED'		
1.2 TOTAL-PERS		
PERS-CNTR = (N)		

PROGRAM NAME: PROBLEM SOLVING PE #3STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 2040MODULE NAME: FINISH-UPPAGE 1 of 1

356

INPUT	PROCESS	OUTPUT
1. OPENED PERSONNEL- MASTER-FILE	CLOSE PERSONNEL-MASTER-FILE. CLOSE PRINT-FILE.	1. CLOSED PERSONNEL- MASTER-FILE
2. OPENED PRINT-FILE	EXIT.	2. CLOSED PRINT-FILE

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

IA-01-03-26

6



PROGRAM NAME: PROBLEM SOLVING PE #3STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 3020MODULE NAME: WRITE-CONTROLPAGE 1 of 1W
S
O

INPUT	PROCESS	OUTPUT
	IF LINE-CNTR IS GREATER THAN 19 THEN	
	PERFORM 4010-WRITE-HEADER	
	MOVE 1 TO LINE-CNTR	
	ELSE NEXT SENTENCE.	
	PERFORM 4020-WRITE-DETAIL.	
	ADD 1 TO LINE-CNTR.	
LINE-CNTR = (N)	EXIT.	LINE-CNTR = (N+1) OR 1

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

IA-01-03-26

4

PROGRAM NAME: PROBLEM-SOLVING PE #3MODULE NUMBER: 3A3ASTUDENT NAME: KUCINICHMODULE NAME: READDATE: FEB 79PAGE 1 of 1

IPO

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER- FILE	READ PERSONNEL-MASTER-FILE AT-END	1. PERSONNEL-MASTER- RECORD
	MOVE 'ON' TO EOF-SW.	
	EXIT.	
EOF-SW = 'OFF'		EOF-SW = 'OFF' OR 'ON'

360

INPUT	PROCESS	OUTPUT
	WRITE HEADER-LINE AFTER ADVANCING TO TOP	1. HEADER-LINE
	OF PAGE	
	EXIT.	
1. HEADER-LINE		
1.1 'SSN'		
1.2 'NAME'		
1.3 'RANK'		
1.4 'NR DEPS'		



PROGRAM NAME: PROBLEM SOLVING PE #3

STUDENT NAME: KUCINICH

DATE: FEB 79

IPO

MODULE NUMBER: 4020

MODULE NAME: WRITE-DETAIL

PAGE 1 of 1

[illegible]

"FOR INSTRUCTIONAL PURPOSES ONLY"

TA-01-03-2.6

66

PROBLEM SOLVING PE #4

I INTRODUCTION.

The Battalion Headquarters was pleased with the report produced from PE #3 but feel they could better use the report with the following modifications.

1. Print only those records of personnel above the rank of PV1.
2. Print an additional total line two lines below the one now being generated which contains a count of the number of PV1's on the Personnel Master File.

Format of second total line is as follows;

TOTAL NUMBER OF PV1 999

PROGRAM NAME: PROBLEM SOLVING PE #4

IPO

MODULE NUMBER: 2030STUDENT NAME: KUCINICHMODULE NAME: WRITE-TOTALDATE: FEB 79

PAGE ____ of ____

INPUT	PROCESS	OUTPUT
	MOVE PERS-CNTR TO TOTAL-PERS.	1. TOTAL-LINE
	WRITE TOTAL-LINE AFTER ADVANCING 2 LINES.	2. TOTAL-LINE-2
	MOVE PV1-CNTR TO TOTAL-PV1.	
1. TOTAL-LINE	WRITE TOTAL-LINE-2 AFTER ADVANCING 2	
1.1 'TOTAL RECORDS	LINES.	
PRINTED'		
1.2 TOTAL-PERS		
2. TOTAL-LINE-2		
2.1 'TOTAL NUMBER OF		
PV1'		
2.2 TOTAL-PV1		
PERS-CNTR = (N)		
PV1-CNTR = (N)		

PROGRAM NAME: PROBLEM SOLVING PE #4STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 2020MODULE NAME: COUNT-PROCESSPAGE 1 of 1

364

INPUT	PROCESS	OUTPUT
1. PERSONNEL-MASTER-	IF PMR-RANK = 'PV1'	
RECORD	THEN ADD 1 TO PV1-CNTR	
1.1 PMR-RANK	ELSE ADD 1 TO PERS-CNTR	
	PERFORM 3010-MOVE	
	PERFORM 3020-WRITE-CONTROL.	
	PERFORM 3030-READ.	
PERS-CNTR = (N)	EXIT.	PERS-CNTR = (N) OR (N+1)
PV1-CNTR = (N)		PV1-CNTR = (N) OR (N+1)

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

2

IA-01-04-01



2

PROGRAM NAME: PROBLEM SOLVING PE #4STUDENT NAME: KUCINICHDATE: FEB 79

IPO

MODULE NUMBER: 2019MODULE NAME: GET-READYPAGE 1 of 1

INPUT	PROCESS	OUTPUT
1. CLOSED PERSONNEL- MASTER-FILE	OPEN INPUT PERSONNEL-MASTER-FILE.	1. OPENED PERSONNEL- MASTER-FILE
2. CLOSED PRINT-FILE	OPEN OUTPUT PRINT-FILE.	2. OPENED PRINT-FILE
	MOVE 99 TO LINE-CNTR.	
	MOVE 'OFF' TO EOF-SW.	
	MOVE 0 TO PERS-CNTR.	
LINE-CNTR	MOVE 0 TO PV1-CNTR.	LINE-CNTR = 99
EOF-SW	PERFORM 3030-READ.	EOF-SW = 'OFF'
PERS-CNTR	EXIT.	PERS-CNTR = 0
PV1-CNTR		PV1-CNTR = 0

121-013-1401-080-A

"FOR INSTRUCTIONAL PURPOSES ONLY"

3

IA-01-04-01

USAFPP 92-7 4/79

365

SORT EXERCISES

1. Code the jobstream to sort a file. I/O specifications follow (Use 1 sort work area):

	<u>Input</u>	<u>Output</u>
Media:	Tape	Tape
Rec Length:	80/1600	80/1600
File ID:	PERS-TX	SORTED-PERS-TX
Volume of Data	500 records	500 records
Device:	TP0	TP1
Control Field	pos 8-15 (character data)	

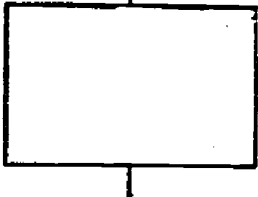
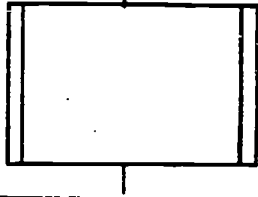
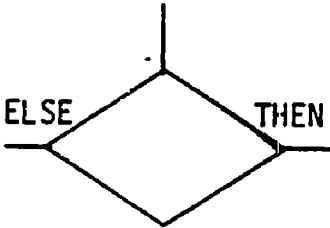
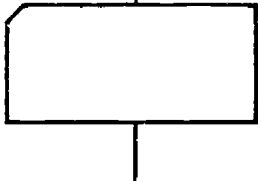
2. Assume that you have 3 sorted files in the format of problem 1. Code the jobstream to merge them.

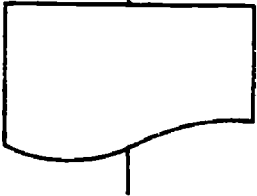
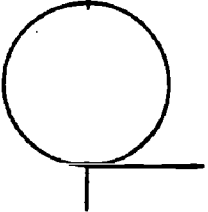
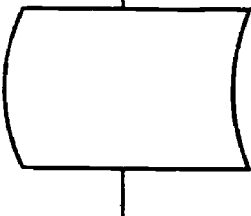
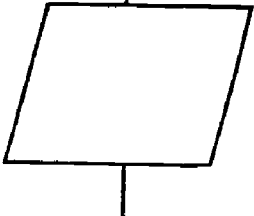
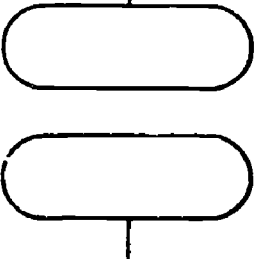
3. Code the jobstream to sort the following files (Use 2 sort work extents):

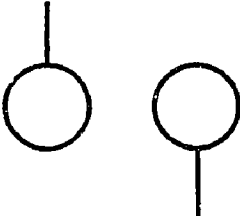
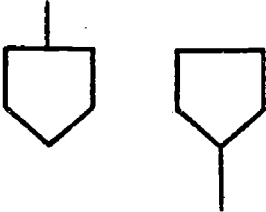
	<u>Input1</u>	<u>Input2</u>	<u>Output</u>
Media:	Tape	Tape	Disk
Rec length/Blksize:	100/1000	100/1000	100/7000
File ID:	LOG-MSTR	LOG-TEMP	SORTED-LOG-MSTR
Volume of data:	20000 records	30000 records	50000 records
Device:	TP1	TP0	DK4
Volume Serial Nr:	N/A	N/A	LOG PAK
Retention:	N/A	N/A	3 months
Control fields:			
MAJOR:	8-20, char		
INT 1:	29-30, char		
INT 2:	24, char		
MINOR:	30-36, char		

STRUCTURED FLOWCHARTING SYMBOLS

The flowcharting symbols defined in this handout are to be used in conjunction with the Verbs defined in the COBOL SOURCE DESIGN LANGUAGE handout. This handout does not contain all of symbols that can be used in structured flowcharting but is limited to those symbols that will be used in the problem solving block of instruction.

SYMBOL	USE
<p>PROCESS SYMBOL</p> 	<p>The process symbol may be used to represent the ADD, CLOSE, OPEN, MOVE or NEXT SENTENCE. This symbol requires one entry and one exit.</p>
<p>PREDEFINED PROCESS SYMBOL</p> 	<p>The predefined process symbol may be used only to represent a PERFORM. This symbol requires one entry and one exit.</p>
<p>DECISION</p> 	<p>The decision symbol may be used only to represent the IF-THEN-ELSE. This symbol requires one entry and two exits, one THEN exit and one ELSE exit.</p>
<p>PUNCHED CARD</p> 	<p>The punched card symbol may be used to represent a READ for an input punched card file or a WRITE of a punched card record. When used for a read it will have one entry and two exits. When used for a write it will have one entry and one exit.</p>

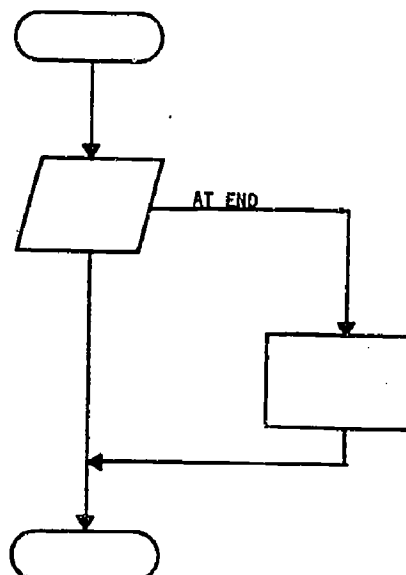
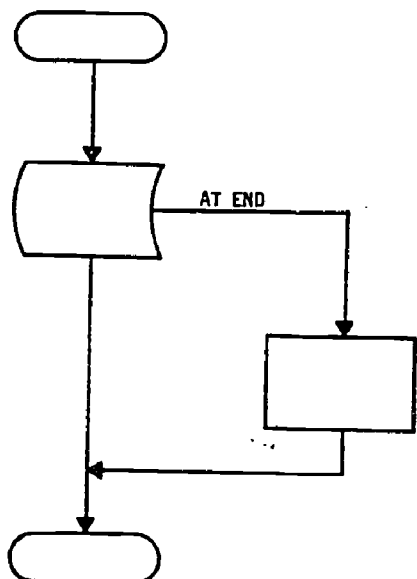
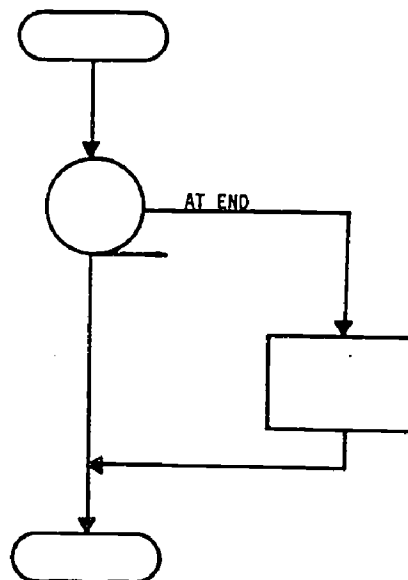
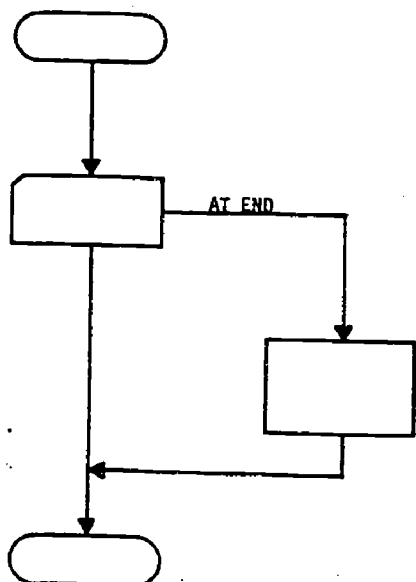
SYMBOL	USE
<p>DOCUMENT</p> 	<p>The document symbol may be used only to represent a WRITE to a printer. This symbol requires one entry and one exit.</p>
<p>MAGNETIC TAPE</p> 	<p>The magnetic tape symbol is used in the same way as the punched card symbol except where the storage media is magnetic tape.</p>
<p>ONLINE STORAGE</p> 	<p>The online storage symbol is used in the same way as the punched card symbol except where the storage media is a disk.</p>
<p>INPUT/OUTPUT</p> 	<p>The input/output symbol is used in the same way as the punched card symbol except where the storage media is unknown.</p>
<p>TERMINAL</p> 	<p>The terminal symbol is used to show the beginning and ending point of each routine. It will contain the module number and name or the word EXIT or STOP RUN. This symbol requires either one entry or one exit but not both.</p>

SYMBOL	USE
ON PAGE CONNECTOR	The on page connector symbol is used to connect two parts of the same routine which are drawn separately on the same sheet of paper.
	
OFF PAGE CONNECTOR	The off page connector symbol is used to connect two parts of the same routine which are drawn separately on separate sheets of paper.
	

When using the PUNCH CARD, MAGNETIC TAPE, ONLINE STORAGE or INPUT/OUTPUT symbol for a READ, you will always show the AT END associated with it as follows: (See page 4)

For an illustration of the use of an IF-THEN-ELSE and a compound IF-THEN-ELSE see page 5.

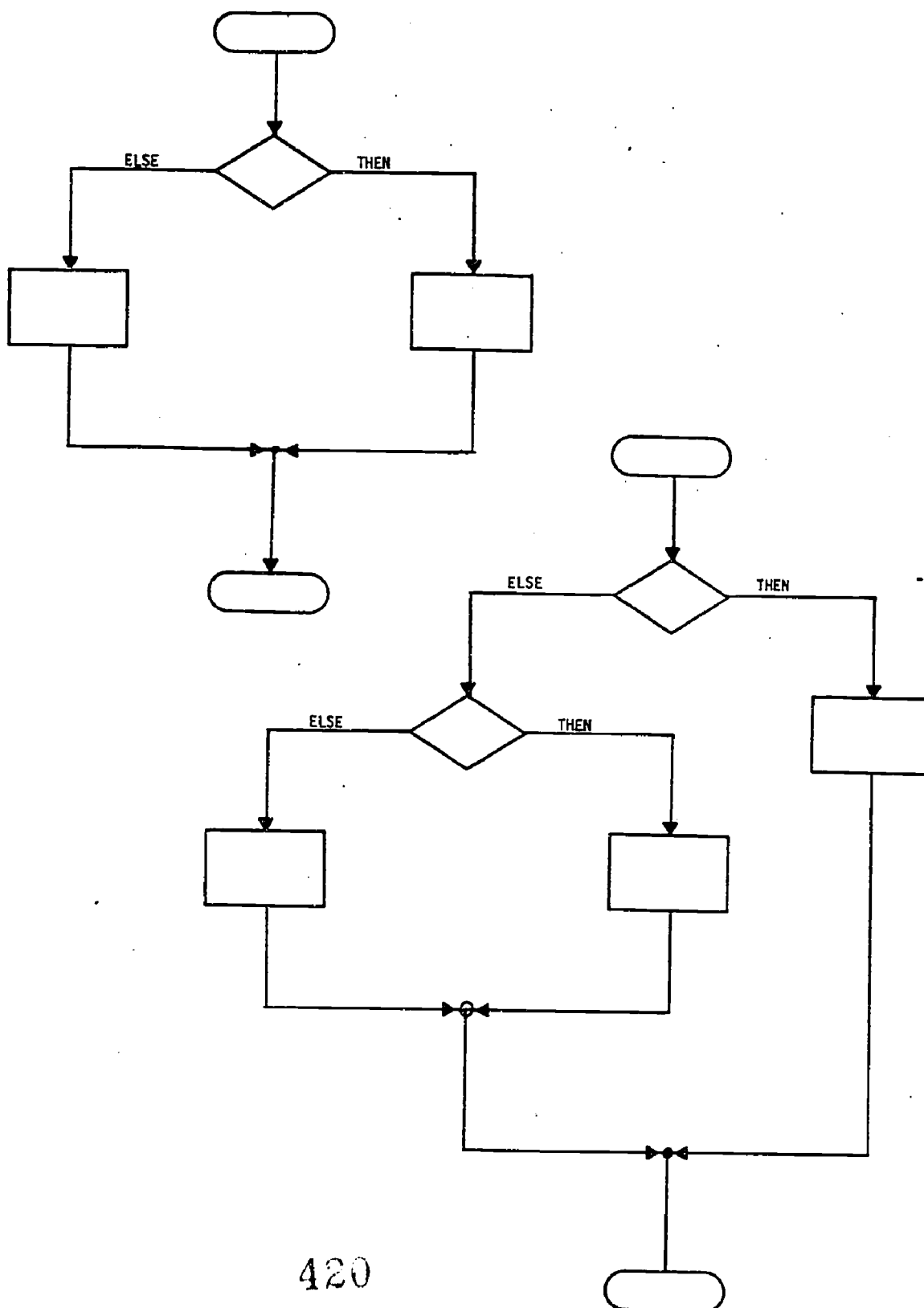
Programmer: Kucinich Program No.: _____ Date: Jan 79 Page: 4
 Chart ID: _____ Chart Name: READ routine structured flowcharts Program Name: _____



IBM Flowcharting Worksheet

FOR INSTRUCTIONAL PURPOSES ONLY

Programmer: Kucinich Program No.: _____ Date: Jan 79 Page: 5
 Chart ID: _____ Chart Name: IF-THEN-ELSE structured flowchart Program Name: _____



420

UTILITY EXERCISES

1. Code the jobstream to copy a deck of cards to tape. Columns 21 and 30 must be interchanged on the output tape, additionally columns 71-80 must be switched with columns 1-10. All other columns remain the same. The output tape must have 20 records per block and should be unloaded at EOJ.
2. Code the jobstream to list a file on disk. The file is on tracks 80-200 on pack DATA19. The records on disk are 100 bytes long and blocked 20 records per block. The listing should be single spaced. A 2314 disk will be used. The file-ID of the disk file is 'MOD WORKOUT'.
3. Write the jobstream to punch cards from a disk file. The file is called 'PAY RECORDS' and is on pack PAYPAC (omit extent info). The records are 120 bytes blocked by 20. The first 40 and last 40 bytes make up the 80 columns of output data. The first 1500 records should be bypassed.
4. Write the jobstream to print the cards punched in problem 3. All 80 bytes are to be printed with the output having 5 spaces between each group of 40 bytes (i.e.: 40 bytes 5 spaces, 40 bytes). The listing should be double spaced with page numbers.
5. Write the jobstream to load a tape file to disk. The input file-ID is 'A01ABC TRANS' and the record length is 100 blocked by 30. The output file is stored on pack DLOG01 with extents and file-ID determined by the programmer. Output retention is one month.
6. Write the jobstream to copy a tape file. Input tape is 'PAY FILE' created on 6 Jan and is 90 bytes per record blocked by 10. Output should be blocked by 30 and be retained 3 months. Input file is generation 5, output is generation 6.

ANNEX E

121-013-1419-070-A

421
1

IA-01-05-22

UTILITIES P.E.

The PSNCO of your unit has brought you a deck of cards that he wants listed. The format for these cards is attached. The cards represent all enlisted personnel in the battalion. He does not know the current sequence of these cards, but he requires the output to be printed in sequence according to the print record layout below. Code the jobstream that will sort the cards and print them according to the customer's desires.

NOTES:

1. The SORT program will not read cards or print so two utility programs are required; one to transfer the card data to disk and another to print the sorted file.
2. Use 1 sort work file on disk.
3. Use ADAS JCL for all disk files (Volume serial no = SYSWRK).

<u>Field</u>	<u>Sequence</u>	<u>Card</u>	<u>Print</u>
Social Security Number		1-9	N/A
Name	Minor; A	11-30	21-40
MOS		32-38	42-48
Grade	Int; D	40-42	50-52
Company	Major; D	44	14
Date of Rank		46-51	56-61

**DISK OPERATING SYSTEM
INTERFILE TRANSFER AND MANIPULATOR PROGRAM
FOR AID IN TESTING AND OPERATIONS
"DITTO"**

CSD.SFW.DITTO

TABLE OF CONTENTS

I. Title Page	1
II. Table of Contents	2
III. Program Abstract	3
IV. Description of Functions	4
V. Control Card Operation	10
VI. Console Operating Instructions	12
VII. Multiprogramming Considerations	13
VIII. Table of Function Codes	15
IX. Operating Suggestions	17
X. Implementation	19
XI. Tape and Deck Key	20
XII. Sample Programs	21

Ditto is used throughout this documentation in lieu of the program title, "DOS Interfile Transfer and Manipulator Program For Aid In Testing and Operations."

PROGRAM ABSTRACT

DOS Interfile Transfer And Manipulator Program For Aid In Testing And Operations - (DITTO), is a generalized program which operates in a DOS environment. Thirty-three (33) functions are available to support Unit Record, Tape and 2311 and 2314 Disk. Functions available in addition to normal tape, disk and card functions are: Tape and Disk Record Scan, Disk and Tape Record Alteration, Disk ID Volume Number Change, Initialize Tape, Deblocking of Tape and Disk Records When Printing, and User Tape Error Handling.

Operating Characteristics Are:

1. The program is self-relocating to allow operating in any partition. Tape and Disk record size is limited only by the amount of core available at execution time.
2. Operations may be performed entirely from the console or function card statements may be job-streamed for remote programmer use.
3. All console communication is in terms of physical hardware I/O addresses, eliminating the need for knowledge of current logical assignments. Tape density and mode settings may be entered via the console.
4. DITTO functions in a DASD file protected environment without volume label statements. Printed output reflects the physical and logical addresses and file characteristics (density, etc.) of the input file.
5. Minimum requirements: System/360 DOS System with a 1403 Printer and a minimum 14K problem program area. The program is written in Assembler language.

DESCRIPTION OF FUNCTIONSDisk Print

2311 or 2314 Disk to Printer in character and vertical format within specified limits. Disk records are "separated" into key and data format and printed with their associated cylinder, head, and record addresses. When a defective track is detected, the assigned alternate track with its respective data is printed. This provides a sequential listing of the disk file in "logical" sequence.

Two Disk to Printer formats are available: Disk Dump Unblocked; and Disk Dump with Reblocking. The Unblocked format lists the actual records as they exist on the file. The Reblock function allows the user to submit a logical record size. This size parameter should be equivalent to the actual logical record length only, and should not include separate key length. Imbedded keys should, however, be included. The Reblock function lists the record key and then deblocks the data portion of the record into logical lengths. This

CSD.SFW.DITTO

provides a more easily used listing of large physical record files.

The Reblock function is primarily intended for use on sequential or random files. Because of the different formats present on index sequential files, reblocking may not be meaningful for overflow areas or cylinder and track indices.

Disk Record Scan

Disk Record Scan provides the capability of scanning a disk file and recording the locations of all records which match a given scan argument. Three (3) scan argument types are available.

"Scan on Key" allows the user to scan for a match on any portion of the disk record key fields. A 1 to 35 position scan argument, and its associated starting position within the key field, are entered via the console. Beginning and ending disk limits are also entered. DITTO will scan the specified disk area and log all "hits" in the form of cylinder, head and record numbers on SYSLOG.

"Scan on Data" provides a similar capability for the data portion of disk records. A logical record length must be specified to allow internal de-blocking for scanning blocked files.

"Scan on EOF" will log all End of File record locations. Data entry for "EOF" is not required.

Disk Record Load

This function will alter the contents of the key and/or data portion of an existing 2311 or 2314 Disk Record. After the required disk unit and record addresses have been entered, the record will be retrieved and "separated" into key and data format and printed on SYSLST.

At this point, DITTO will request the number of bytes to be changed and the field type (Key or Data portion of the record) to be altered. After specifying these parameters, a starting position within the field will be requested. This starting position is based on using the first position of the key or data as position "0001". The user can verify this position by referencing the printout of the record and its associated scale on SYSLST. Data may be entered via the console in character (1 character per byte of change), or hexadecimal (2 characters per byte of change). Valid hexadecimal "characters" are 0-9 and A through F.

After all alterations to the record in core have been made, the altered record will again be printed on SYSLST and a message issued "ARE CHANGES COMPLETE - Y OR N." A response of "N" (NO) will allow the user to re-enter the

alter routine. A response of "Y" (YES) will rewrite the record onto disk.

Write Disk End-of-File Record

This function will retrieve any existing 2311 or 2314 Disk Record and re-write the record as an end-of-file record (key and data length of zero). The user must supply the cylinder, head, and record address of the record to be altered.

Split Cylinder Disk Organizations

Split cylinder data file organizations are supported by DITTO for the Disk Dump Unblocked (DDU), Disk Dump Reblocked (DDR), and the Disk Record Scan (DRS) functions. Split cylinder operations are denoted by replacing the first D in the above function codes with an S. For these split cylinder operations, the beginning and ending head addresses will determine the lower and upper head boundaries for the function.

Disk Identification Change

This function will print on SYSLOG the existing disk volume serial number and allow the user to enter on SYSLOG a new volume serial number.

Initialize Tape

Initialize tape using the standard DOS format with a tape volume serial number entered via the console. Existing tape labels or data on the output tape is not checked before creating the new label set.

Tape To Tape

Tape-to-tape copy including tape marks. This function will copy from 1 to 99 tape files per volume as requested via the console. One file will be copied per control card if console operation is not being used. The first file copied may or may not have a leading tape mark. A leading tape mark does not itself constitute a separate file. Block count statistics will be printed on SYSLST for control card operation, and on SYSLOG for console operation, for each tape file copied.

Tape Record Load

This function is used to copy and selectively alter existing tape records. The user need not know the exact location (block #) of the input tape record to be altered.

The first portion of the Tape Record Load function locates the desired record. The user enters an estimated block location and the parameter "F" to denote copying of tape in a forward position. When the desired record is found, it is printed on SYSLSI and an option given to alter the record. All input records prior to the record to be altered have been copied onto the output tape. If the printed record is not the desired record, the user can again enter the # of blocks and direction to reposition the input and output tapes. If the user has gone past the desired record, a parameter of "B" should be entered with a block count. This will cause both input and output tapes to be backspaced the required number of blocks. This count includes the block just printed. When the desired record is located, the user may alter the length and/or data content in a manner similar to the Disk Record Load function. After all alterations are completed, replies of "9999" and "F" (9999 blocks to be copied in a forward direction) will usually be sufficient to finish copying the tape file.

Tape Record Scan

This function allows the user to specify a logical record size and a 1 to 35 position scan argument for an input tape. DITTO will read and internally deblock input records until the desired record is located. At that time the entire physical record with its associated block number will be printed and the operation terminated. The tape will not be rewound.

Tape Print

Four printing formats are available for input tapes. Tape data may be printed in character format only, or in character and vertical hex format. Physical tape records may be printed as they exist (unblocked) or may be deblocked into logical records. Block and record counts are printed with the associated records.

For control card operation, an optional NBLKS=nnnn parameter may be specified. "nnnn" specifies the maximum number of physical tape record blocks to be printed. If "nnnn" is greater than the existing number of blocks in the tape file, only the existing tape file will be printed. A leading tape mark will be automatically skipped.

Print SYSLST Tapes

Tape to printer unblocked using the first position of the tape record as the carriage control character. Two carriage control formats are available. Type A forms control (TFA) uses the carriage control character as the CCW operation code. DITTO performs its printing using Type A. Type D forms control (TFD) is used by DOS for compiler listings, etc.

Tape Control

Eight tape control functions are available either by control cards or console communications. These control functions are: write tape mark; rewind tape; rewind and unload tape; forward space or backspace file; forward space or backspace records; and erase record. The erase record will remove approximately 4" of tape data.

Card To Tape

Card image to tape with or without reblocking. Job control cards, including /* and /& cards, may be interspersed in the input data (see Card Input Assignment). A tape mark will be written after the last card block on the tape file.

Tape To Card

Tape to card for blocked or unblocked 80 character logical records. A leading tape mark will be bypassed if present. If the input tape record is not a multiple of 80 bytes, the function will terminate.

Card To Printer

Cards may be listed on SYSLST in either character or character and vertical hex format. Job Control cards including /* and /& statements may be interspersed in the input data (see Card Input Assignment).

Card To Card

Two card to punch options are available. CCU is an 80/80 reproduction of the input cards. CCS is a card-to-card copy with sequence numbers and deck identification name added. Three decktypes are available.

- DECKTYPE=COB is used to sequence and name COBOL source decks. Sequence numbers will be placed in CC 1-6 and a 0-8 position deckname placed in CC 73-80.
- DECKTYPE=RPG is used for RPG source decks. Sequence numbers will be placed in CC 1-5 and a 0-6 position deckname placed in CC 75-80.
- DECKTYPE=BAL is used for Assembler and FORTRAN Source decks, Job Control decks, etc. Sequence numbers will be placed in CC 77-80 and a 0-4 position deckname placed in CC 73-76.

Decknames will be left-justified and padded with blanks if less than the required length. /* and /& cards may be interspersed in the input data (see Card Input Assignment).

End Of Job

All functions, whether control card or console initiated, must issue the function code "EOJ" to return control to Job control. For control card operations, this "EOJ" card must be the last function requested.

Tape Errors

All tape errors will initially be handled by the DOS supervisor. For input tape errors that are uncorrectable, DITTO will transfer control to a console routine. This routine will display the record as read, and allow the operator to accept, bypass, or alter the data and/or length of the tape record before returning control to the original operation. This option is not available for remote programmer control card operation.

Card Input Assignment

If the user does not desire to include /* and/or /& cards as a part of his input data, normal DOS procedures should be followed. For control card operations, the end of the input deck is signaled with a /* card. This will denote end-of-file for the particular operation and is consistent with DOS operating procedures. The /* card may be followed with additional DITTO operations in a similar manner. Operations which do not require data card input should not contain a /* card. The last DITTO operation must be an EOJ card and should be followed by a /* card if other DOS job steps follow, or by a /& card if a new job is to be initiated by Job Control. Console operations for data card input may be performed in a similar manner. For console operations which do not have a stacked job input stream, /* and /& cards may be omitted. End-of-file on card input will be denoted by physical end-of-file on the input device.

The DOS Supervisor monitors all cards read by the problem program from SYSIPT or SYSRDR. If the problem program attempts to read past a /& statement using either of these system logical units, the Supervisor cancels the job. The Supervisor does not, however, monitor data from a programmer logical unit. To read /& statements, DITTO must, therefore, read card input data from a programmer logical unit.

The inclusion of /* and /& cards in the input is available under console operation only. To initiate a card input operation with /* and/or /& cards in the input data, the operator must substitute an "&" for the first "C" in the function code (i.e. CCS becomes &CS). A blank card in the input stream denotes end-of-file.

CONTROL CARD OPERATION

If the user wishes to remove the decision requirements from the console operator, he may prepare control cards and submit the task in a normal Job Stream environment.

To denote control card operation the user must submit between the Job card and Exec card a // UPSI 1 card. DITTO will test the communications region to determine whether control card or console operation is desired. Depressing the interrupt key at any time overrides the control card option.

The required control card information can be contained on one card. Each control card is treated as a new operation. The control card format is as follows:

```
CC 1  -7      $$DITTO
CC 10 -12     Function Code (see table)
CC 16         Parameter 1,.....,Parameter N
```

Parameters are in standard key word format. Each parameter must be separated with a comma with no embedded blanks. A blank stops the card scan. Lower case letters denote user implied information. Parenthesis () denote optional parameters. Refer to the "Parameter Requirements" for a listing of parameters associated with each function.

<u>Parameter</u>	<u>Description</u>
INPUT=SYSnnn	Logical input device
OUTPUT=SYSnnn	Logical output device
BEGIN=ccchh	Lower disk extent
END=ccchh	Upper disk extent
NBLKS=nnnn	Number of tape blocks
RECSIZE=nnnnn	Logical record size
BLKFACTOR=nnn	Output blocking factor (CTR)
DECKTYPE=xxx	CCS decktype
DECKNAME=x....x	CCS deckname (0-8) characters. Decknames which are less in length than required will be left

CSD.SFW.DITTO

justified and padded with blanks.

PARAMETER REQUIREMENTS

<u>CC 1-7</u>	<u>CC 10-12</u>	<u>C 16</u>
\$\$DITTO	DDU	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh
\$\$DITTO	DDR	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh,RECSIZE=nnnnn
\$\$DITTO	SDU	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh
\$\$DITTO	SDR	INPUT=SYSnnn,BEGIN=ccchh,END=ccchh,RECSIZE=nnnnn
\$\$DITTO	TPU	INPUT=SYSnnn(,NBLKS=nnnn)
\$\$DITTO	TPR	INPUT=SYSnnn,RECSIZE=nnnnn(,NBLKS=nnnn)
\$\$DITTO	THU	INPUT=SYSnnn(,NBLKS=nnnn)
\$\$DITTO	THR	INPUT=SYSnnn,RECSIZE=nnnnn(,NBLKS=nnnn)
\$\$DITTO	TFA	INPUT=SYSnnn
\$\$DITTO	TFD	INPUT=SYSnnn
\$\$DITTO	TCR	INPUT=SYSnnn
\$\$DITTO	CTU	OUTPUT=SYSnnn
\$\$DITTO	CTR	OUTPUT=SYSnnn,BLKFACTOR=nnn
\$\$DITTO	TTU	INPUT=SYSnnn,OUTPUT=SYSnnn
\$\$DITTO	WTM	OUTPUT=SYSnnn
\$\$DITTO	REW	OUTPUT=SYSnnn
\$\$DITTO	RUN	OUTPUT=SYSnnn
\$\$DITTO	FSF	OUTPUT=SYSnnn
\$\$DITTO	BSF	OUTPUT=SYSnnn
\$\$DITTO	FSR	OUTPUT=SYSnnn,NBLKS=nnnn
\$\$DITTO	BSR	OUTPUT=SYSnnn,NBLKS=nnnn
\$\$DITTO	CPU	
\$\$DITTO	CHU	

<u>CC 1-7</u>	<u>CC 10-12</u>	<u>C 16</u>
\$\$DITTO	CCU	
\$\$DITTO	CCS	DECKTYPE=xxx,DECKNAME=xx...x
\$\$DITTO	EOJ	

CONSOLE OPERATIONS

All operator communication may be in upper or lower case. The cancel key is also implemented allowing retyping of replies in error. The end of message may be denoted by the "EOB" key or by repeatedly depressing the space bar.

Depressing the interrupt key for background operation will cause the existing function to be terminated and a new operation request displayed on the console. (See Multiprogramming Considerations for interrupting foreground operation.)

Tape address replies are entered in the form of "CUUMM" where C=channel, UU=unit address, and MM=the density and mode setting. For nine-track single density tapes, blanks or zeros may be substituted for C0, or the MM parameter may be omitted.

If the operator desires the inclusion of /* and/or /& cards in the input data for card input functions (i.e. loading Job Streams on tape, etc.), he must replace the first "C" of the function code with a "&". For these operations, a blank card denotes end-of-file. If the operator desires to cancel a program with card data input, he should interrupt DITTO and enter "CCL" on the console. This will cause the input stream to be flushed up to and including the end-of-file card.

DITTO may also be executed in the background entirely from the console by entering:

```
// JOB Anyname
// EXEC DITTO
```

After completing the desired operations, including the function "EOJ", a "READY FOR COMMUNICATIONS" message will be printed by Job Control. Reply:

```
/&
```

MULTIPROGRAMMING CONSIDERATIONS

Interrupting Foreground Operations

To interrupt DITTO when executing in a background partition, the external interrupt key on the CPU is depressed. This will cause the existing DITTO

CSD.SFW.DITTO

11

function to be terminated and a console message issued requesting a new function.

The interrupt DITTO when executing in a foreground partition, the following procedures should be used:

1. Depress Console Request Key.
2. Message "AR READY FOR COMMUNICATIONS" will be displayed.
3. Enter "MSG Fn", where n=1 or 2 and correspond to the partition being interrupted.
4. Enter EOB (Alt. coding 5).

The existing foreground DITTO function will be terminated and a console message issued requesting a new function.

Batch Job Foreground Operation (DOS Version III Only)

Requirements for Batch Job Foreground operation are identical to those of background operations. Control card operations may be interchanged in any batch partition. System units (SYSLST, SYSIPT, SYSPCH) must be assigned if used in any of the DITTO functions.

Foreground Initiator Operation

Foreground Initiator Operation of DITTO is available in the console mode of communication only. Control card operation, due to the lack of an active communications region for the partition, is not available. Due to the lack of system units (SYSIPT, SYSPCH, SYSLST) in the foreground for DOS Version II, SYS001, SYS002 and SYS003 should be assigned for the above devices respectively. These system unit restrictions do not apply to DOS Version III users.

System Unit Assignments

DITTO will determine at load time the device types assigned to SYSLST and SYSPCH. If SYSLST is assigned to tape, 133 character records will be written with the first character being the Type A carriage control character. If SYSPCH is assigned to tape, 80 character records will be written. At EOJ, a tape mark will be written and the tape backspaced for the appropriate System Unit if data has been written.

If SYSPCH is assigned to a unit record device, the appropriate write command for either a 2500 series, 1442 N1, or 1442 N2 punch will be used based on the user's supervisor. No DITTO modifications should be made for these devices.

FUNCTION CODES

<u>Code</u>	<u>Alternate</u>	<u>Description</u>
CCU	CC	Card to Card
CCS	--	Card to Card with sequence numbers and deck name identification
CPU	CP	Card to Printer in character format
CHU	CH	Card to Printer in character and vertical hexadecimal format
CTU	CT	Card to Tape Unblocked
CTR	--	Card to Tape Reblocked
TCR	TC	Tape to Card unblocked or blocked
TPU	TP	Tape to Printer unblocked in character format
TPR	--	Tape to Printer reblocked in character format
THU	TH	Tape to Printer unblocked in character and vertical hexadecimal format
THR	--	Tape to Printer reblocked in character and vertical hexadecimal format
TTU	TT	Tape to Tape (1 to 99 tape files)
*INT	--	Initialize Tape - DOS standard format
*TRL	--	Tape Record Load. Copy tape to tape with altering of length and/or data of specific records
*TRS	--	Tape Record Scan. Search tape for logical record using 1-35 position scan argument
TFA	--	Tape to Printer using Type A forms control
TFD	--	Tape to Printer using Type D forms control
WTM	--	Write Tape Mark
REW	--	Rewind Tape
RUN	--	Rewind and Unload Tape
FSF	--	Forward Space tape file

CSD,SFW,DITTO

<u>CODE</u>	<u>ALTERNATE</u>	<u>DESCRIPTION</u>
BSF	--	Backspace tape file
FSR	--	Forward Space tape record
LSR	--	Back space tape record
ERG	--	Erase Record Gap
DID	—	Disk Identification change (change volume serial number)
DRL	--	Disk Record Load. Alter key and/or data portion of any disk record.
DDU	DD	Disk Dump. Disk to printer in character and vertical hexadecimal format unblocked.
DDR	--	Disk Dump with reblocking of data records into logical record length.
DRS	--	Disk Record Scan. Scan disk for matching key or data argument or for EOF.
SDU	SD	Disk Dump for split cylinder files
SDR	--	Disk Dump reblocked for split cylinder files
SRS	--	Disk Record Scan for split cylinder files
EOF	--	Write disk end-of-file record
CCL	--	Cancel card input function
EOJ	--	End of Job

* Denotes functions not available with control card operation.

OPERATING SUGGESTIONS

DEBLOCKING TAPES

DITTO does not provide a tape to tape with deblocking function. Blocked card image tapes (i.e., IBM supplied DTR tapes, DOS maintenance tapes, etc.), may be deblocked into an unblocked format, however, by using the tape to card function and assigning SYSPCH to the output tape. The deblocked tape may then be assigned to SYSIPT, eliminating the need in many cases of physically punching the data into cards.

CSD,SFW,DITTO

PRINTING TAPE FILES

Programmers who require listings of tape files will find the Tape Hex formats more useful than the character print functions. The Tape Hex functions provide both character and vertical hexadecimal format with a position scale for each tape record. Programmers using these functions under the remote control card operation should also make use of the NBLKS parameter. This optional parameter gives the programmer the capability of obtaining "partial" listings of tape files. This eliminates "lengthy" and "time consuming" tape printing.

PRINTING SYSLST TAPES

When SYSLST is assigned to tape, DOS Job Control statements are written as 121 character records, with the first character being the ASA FORTRAN forms-control character (Type D). DITTO and many other Type III programs which support SYSLST on tape, write 133 character records, with the first character being the CCW operation code for forms-control (Type A). Printing SYSLST tapes presents, therefore, some incompatibilities in forms-control characters to be used.

DITTO provides two (2) functions for printing SYSLST tapes using forms-control characters. TFD assumes all tape records contain the Type D forms-control character. These tape records may contain from 0 to 132 positions of data to be printed.

10. WHAT IS AN INDEXED SEQUENTIAL FILE? Before answering that question, it may be necessary to define a couple of terms. They are physical organization, and logical organization. Physical organization refers to the specific physical layout and placement of records within a file from the standpoint of the hardware. Logical organization, on the other hand, has to do with how the programmer or the operating system for that matter, "sees" the records within a file. Physical and logical organization may be exactly the same for any one file, but that is usually not the case when dealing with files which are resident on tape or on a direct access device.

An indexed sequential (ISAM) file is one in which the records are in sequence logically but not necessarily physically. It provides sequential processing capability just as any other sequential file, and in addition, provides the ability to directly access individual records. It must be resident on a direct access storage device (e.g., disk). It differs from a sequential file in that each record within the file contains a specific control field called a key. This key field must be identified to the operating system, and the key within each record must be unique; i.e., no two records in the file can have the same key, just as no two people should have the same social security number.

The total file is composed of at least three separate areas - an index area, a prime data area, and an overflow area. Though these areas are "controlled" by the operating system, the programmer can vary the size of each, and, under certain conditions, can look at what is contained in each. Normally, he is not concerned with the details of these areas.

To create or load an indexed sequential file, the records must be in ascending sequence by key. As the records are loaded, the system builds a series of indexes, and places all of the records in the prime data area. Nothing goes into the overflow area during the create process. That area is used only when records are subsequently added to the file, and the system needs space for expansion.

The major advantage to using an indexed sequential file is the great amount of flexibility it provides. When records are added, the whole file does not need to be rewritten as is the case with a straight sequential file. Additionally, as mentioned before, direct accessing and processing of records is possible because of the indexes. Any number of records, in any order, may be read or written without processing the entire file.

Now let us look at the mechanics of creating, updating, and listing indexed sequential files using ANS COBOL. All references are to the sample program attached as Appendix A.

11. REVIEW OF ENVIRONMENT DIVISION AND DATA DIVISION REQUIREMENTS. To process ISAM files, we must describe them properly in the ENVIRONMENT DIVISION and in the DATA DIVISION.

a. ENVIRONMENT DIVISION. SELECT STATEMENT.
(Ref: Lines 54 thru 61.)

(1) The ASSIGN clause.
(Ref: Lines 55 and 58.)

DOS: This clause must be of the form SYSnnn-DA-unit-I in which nnn may be any system number allowed on your system, and unit may be either 2311 or 2314 or 2321. The I identifies the file as being an indexed-sequential file.

OS: This clause must be of the form DA-I-ddname where ddname is the name of the JCL DD card pointing to the indexed-sequential file. This assign identifies the file to the COBOL compiler as being an indexed-sequential file on a direct access device.

(2) The ACCESS clause. When you have an ISAM file, you have the choice of using it as a random file (Direct Access) or as a sequential file. If you wish to use the file as a random file and be able to directly access specific records, you must include the clause ACCESS IS RANDOM in the SELECT statement. If you plan to use the file as a sequential file, you may code either ACCESS IS SEQUENTIAL or omit the ACCESS clause entirely, as SEQUENTIAL is the default. (Ref: Line 59)

(3) The KEY clauses. The RECORD KEY IS clause identifies which field of the ISAM record is the key field. The record key must be specified for any use of an ISAM file. The NOMINAL KEY IS clause is needed whenever ACCESS IS RANDOM is specified. The nominal key is used to identify which record is to be read or written, whereas the record key identifies the one field within the record which is to be used as the key. In other words, when you want a specific record, you must ask for it by its key. This is done by moving the key of the record you want to the nominal key field, and then issuing a READ statement. When accessing an ISAM file sequentially, the nominal key is not required.

(Ref: Lines 56, 60 and 61)

b. DATA DIVISION. FD ENTRIES. The LABEL RECORDS clause in the FD of an ISAM file must be LABEL RECORDS ARE STANDARD. ISAM files may contain only fixed-length records, which may be blocked. As with any other type of file, if the records are blocked, the BLOCK CONTAINS clause must be included as part of the FD.

(Ref: Lines 89 thru 93 and 99 thru 103.)

12. CREATING AN ISAM FILE. (Ref: Lines 208 thru 228) We need two files to create an ISAM file - an input file within which the records, as mentioned before, must be in ascending sequence based on the record key; and an output file which is the ISAM file itself. The Environment Division entry for this file must indicate that access is sequential. The Data Division entry must show LABEL RECORDS ARE STANDARD. In the Procedure Division, the following process is required as a minimum:

- a. The ISAM file must be opened as an output file.
- b. The records read from the input file must be moved to the record area for the ISAM file. If reformatting is desired, it should be done during this step.
- c. To get each record onto the ISAM file, a WRITE statement is needed. The WRITE statement used here is slightly different from that used for straight sequential files. In the event that the records are not actually in sequence, a simple write statement would cause the program to abnormally terminate (ABEND). To prevent this, we can include an INVALID KEY clause with the WRITE statement. The format is as follows:

WRITE record-name FROM data-name
INVALID KEY Imperative Statement(s)

(Ref: Lines 219 thru 226.)

The INVALID KEY is a condition for the WRITE just as AT END is a condition for the READ, and they operate in much the same way. Until the INVALID KEY condition occurs, the statement(s) following the words INVALID KEY are ignored, and control is passed to the next sentence. If the INVALID KEY condition does occur, ONLY then will the statement(s) be executed. The reason for the INVALID KEY option is to prevent us from putting together an invalid file. The INVALID KEY condition will occur whenever we violate one of the three rules concerning building ISAM files.

- Rule 1. The records which will make up the ISAM file must be in ascending sequence by their keys. If we attempt to WRITE a record that is out of sequence, an INVALID KEY condition will occur.
- Rule 2. The key in each record must be unique. If we attempt to WRITE a record whose key is a duplicate of one we have already written, we also will get an INVALID KEY condition.
- Rule 3. Enough disk space must be allocated to contain all records, else an invalid key condition will occur.

The statements following the words INVALID KEY are determined by what you want your program to do when an INVALID KEY condition occurs. You may want to skip over the record that caused the INVALID KEY condition; you may want to print an error message; or you may want to halt the program. Most often, the statements will be a GO TO EXIT or a PERFORM of an error routine.

d. Every record in the file should have a one character field at the very beginning reserved for a deletion code. (Ref: Lines 95 and 105) During creation of the file, the programmer is responsible for moving any non-HIGH-VALUE character to this field (Traditionally LOW-VALUE is moved). Any record in the file with HIGH-VALUES in the first character will not be retrieved when processing the file sequentially.

13. UPDATING AN INDEXED SEQUENTIAL FILE. There are two ways to access an existing ISAM file. Let us look at the mechanics of each.

a. RANDOM ACCESS. Accessing a file randomly means that individual records may be selected from the file in any order. The primary benefit in using this method of processing is that not all records have to be read, and processing time can often be reduced substantially. Considerations for random access:

(1) ENVIRONMENT DIVISION. The SELECT statement for the ISAM file must include the ACCESS IS RANDOM, NOMINAL KEY, and RECORD KEY clauses. (Ref: Lines 57 thru 61.)

(2) DATA DIVISION.

(a) In describing the file (FD), the main consideration is to ensure that the block size, record size, and record format are the same as when the file was created. (Ref: Lines 99 thru 103.)

(b) The data-name associated with the NOMINAL KEY in the SELECT statement must be defined in the WORKING-STORAGE section, preferably as a Level 77 item. It must be the same size, and have the same characteristics as the item within the record which is designated as the RECORD KEY. (Ref: Lines 118, 119.)

(3) PROCEDURE DIVISION. (Ref: Lines 244 thru 287.) Any time an ISAM file is accessed randomly, there must be some means of determining which specific records of the file are to be read. Usually, another input file, called a transaction file, is used to obtain the keys of the records we need. This transaction file generally contains data to update certain records of the ISAM file, and though it may be a physically sequential file, the records

within it do not have to be in any particular order. To handle this type of situation we need to do the following in the Procedure Division:

(a) Open the transaction file as Input.

(b) Open the ISAM file for both Input and Output, because we will read records from that file, update them, and then put them back on the file. Also, we may need to add new records to the file. The OPEN statement looks like this:

OPEN I-O ISAM-file-name

(Ref: Line 176)

(c) A record must now be read from the transaction file. Within that record will be the key of the record to be read from the ISAM file, and it must be moved to the NOMINAL KEY field in working-storage.

(d) The ISAM record may now be retrieved by issuing a READ statement. This particular READ is slightly different from that used with sequential access. Remember, the file is being read randomly, and any record selected may be at the end of the file, in the middle, or at the beginning. In fact, any record may be read more than once in any one processing cycle. So, there is really no end to the number of records read, and an AT END clause is not required with the READ. There is a possibility though, that when we issue a READ for the file, and the system searches for the appropriate record (based on the contents of the NOMINAL KEY), that it may not find that record. If that occurs, an invalid key condition results and unless we provide for handling that condition, the program will ABEND. Fortunately, we can provide for it simply by attaching an INVALID KEY clause to the READ. The format of a READ for random access is:

READ file-name INTO data-name

INVALID KEY statement(s)

(Ref: Lines 250 thru 256.)

The action of the INVALID KEY clause is the same as it was for the WRITE statement; that is, the statement(s) following the words INVALID KEY will not be executed unless an invalid key condition occurs. As stated earlier, that happens when we ask for a particular record, and that record does not exist on the file. It may be that an error of some sort has occurred possibly in source coding or keypunching - and it would then be appropriate to print an error message as well as an image (copy) of the transaction record. Or, if

the processing cycle provides for the addition of new records to the file, it may be that this transaction record is one which should be added, and we would want to perform an add-records routine. The procedures to do that will be discussed later.

(e) Assume for now that upon execution of a READ, no invalid key condition was encountered, and the record we asked for has been retrieved from the file, and is now resident in the ISAM record area of main storage. Any required changes may now be made, and when completed, the record is ready to be written back to the file. In all cases so far, when we needed to put a record onto a file, we used a WRITE statement. In this case however, we must use a REWRITE statement. The reason is that the original record is still on the file. (When we read it, a copy of the record was moved to main storage.) In order to put a "new" copy of the record onto the file, we must REWRITE rather than WRITE. The format of the REWRITE is:

REWRITE record-name FROM data-name

INVALID KEY statement(s)

(Ref: Lines 283 thru 285.)

Notice that we must attach an INVALID KEY clause to the REWRITE. The only reason for an invalid key condition to occur on a rewrite is a change in the contents of the nominal key field between the time the record was read and the time the rewrite was attempted. It is most unlikely that this will happen, but we must provide for it. If it does happen, it is best to print a message indicating what happened (INVALID KEY ON REWRITE), display the record involved, and then terminate the program, as there is evidently a logic error in the program.

(f) In addition to making changes to records, it may be that some records have to be deleted. To indicate which records should be changed, deleted, etc., we can include a type-transaction code in each transaction record. For example, a code of "C" could mean change, a "D" delete, and an "A" add. So far, we have handled only type code "C". The process of deleting records is very similar to that of changing records. The major difference is that we must move HIGH-VALUES to the deletion code field of the ISAM record, which, as pointed out in paragraph 12(d), is a one character field attached to the front of the record at file create time. No other changes need to be made in the record, and it may now be rewritten to the file just as was done with a changed record. Again, (Ref: Lines 270 thru 274) the REWRITE verb must be used instead of WRITE. Though the record is still physically on the file, from a logical standpoint it is not there. It can, however, be retrieved randomly in case it was "deleted" in error.

(g) Now let us look at how we can add new records to an existing ISAM file. An important point we may need to reemphasize here is that the key of each record must be unique. So, if we have to add a new record to the file, it is imperative that there is not already a record on the file with the same key. We can check that in one of two ways. First, let us follow the same procedure as before. We read a record from the transaction file, move the key found there to the NOMINAL KEY area, and then issue a READ to the ISAM file. If an INVALID KEY condition occurs, the record we requested is not on the file. If the type code of the transaction is an "A" then we may add the record to the file using a WRITE statement. (A REWRITE is used only when a record which was previously read has to be put back on the file.) The format of the WRITE is the same as that used in creating the file. Another way to handle the addition of records is to read a record from the transaction file, check the type code to see if it is an "A", and if so, move the transaction record to the ISAM record area, the record key to the nominal key, and then WRITE. If that particular record is not already on the file, the WRITE will be completed successfully (in a very rare case, the file may be full, and an INVALID KEY condition will occur). If there is a record on the file with the same key, an INVALID KEY condition will occur, and it must be handled as an error.

14. SEQUENTIAL RETRIEVAL OF AN INDEXED SEQUENTIAL FILE. (Ref: Lines 305 thru 314) As mentioned before, indexed sequential files can be processed either randomly or sequentially. When updating, it is generally more advantageous to use the random access mode. (To add records to an ISAM file, random access must be used.) Sequential retrieval of records to produce, for example, a printed listing is often necessary. This paragraph will illustrate the process of retrieving sequentially first of all, an entire ISAM file, and secondly, only a portion of that same file. Again let us look at the considerations by COBOL division.

a. ENVIRONMENT DIVISION. The SELECT statement must include the RECORD KEY IS clause following the ASSIGN TO DA-I-sysname clause. The ACCESS IS SEQUENTIAL clause is optional, but should be included for documentation. If it is desired to start the retrieval process at some point other than the first record, then the NOMINAL KEY IS clause must be included also.

b. DATA DIVISION. The only factors to be considered here are that the blocking factor and record size parameters are specified correctly.

c. PROCEDURE DIVISION.

(1) The file must be opened as an input file only.

(2) If the retrieval is to start at a point other than the beginning of the file (first record), the next step must be to move the RECORD KEY of

the first record to be retrieved to the NOMINAL KEY area, and then issue a START command. The format of the START statement is as follows:

START file-name

INVALID KEY imperative statement

This causes the ISAM file to be searched for the record whose key is contained in the NOMINAL KEY area. If the record is found, a pointer is set so that when a READ statement is given (see next paragraph), the first record read will be the one with the given key, and each subsequent issuance of a READ will result in the next record in sequence being read. If the record requested is not found in the file then the statement following INVALID KEY will be executed. Note that the START statement is not required when retrieval is to begin with the first record of the file.

(3) To actually retrieve a record, a READ statement is needed. The format of this particular READ is the same as that for a straight sequential file, i.e.,

READ file-name

AT END imperative statement

(Ref: Lines 306 thru 309)

PE-7 EXAMPLE

COBOL (Structured)

I. INTRODUCTION.

The Supply Officer at Fort Harrison has asked for you to create an ISAM File containing the current Quantity On Hand of the items stocked in the Supply Room. You are to update the ISAM file with each day's Order Records and list the ISAM File after each update.

II. DEFINITIONS.

ISAM FILE; Indexed Sequential Access Method file.

III. INPUT/OUTPUT.

INPUT: Input consists of an Order File stored on a 2314 disk pack. The file contains the Create Records for the ISAM File, followed by a blank record, followed by the days Order Records. Assign this file to SYS006. See Record Layouts 1 and 2 for a description of each type of record.

OUTPUT: Output will consist of two (2) listings.

1. A Transaction Listing showing each transaction as it is processed against the ISAM File with one of the following messages;
 - A. TRANSACTION FOR DELETED RECORD-NOT PROCESSED
 - B. UNMATCHED PART NUMBER
 - C. DELETED RECORD
 - D. NEGATIVE BALANCE ON HAND
 - E. PROCESSED
 See Form Layout #1 for a description of this report.

2. An ISAM FILE LISTING to list all records on the ISAM File by part number showing the updated Quantity On Hand. See Form Layout #2 for a description of this report.

Assign these reports to SYS005.

IV. ASSUMPTIONS.

There will be more than one (1) page of output, so produce a heading at the top of each page. CUSTOMER CODE will also contain a 'D' if the record is to be deleted from the ISAM File. During the updating of the ISAM File no records are to be added.

V. METHODS.

1. Read the CREATE RECORDS and create the ISAM File.
If you run out of space on the disk, or the records are out of sequence, stop the processing at that point.
2. Read the ORDER RECORDS, find the matching ISAM record, and subtract the Quantity Ordered from the Quantity On Hand.
If the record has already been deleted, annotate it with MESSAGE 'A'.
If a matching record is not found, annotate it with MESSAGE 'B'.
If the CUSTOMER CODE is equal to a 'D', delete the record and annotate it with MESSAGE 'C'.
If the updated Quantity On Hand happens to be NEGATIVE, annotate it with MESSAGE 'D'.
Annotate all other transactions with MESSAGE 'E'.

3. List the ISAM File after it is updated.

Both reports will consist of twenty (20) double spaced lines per page.

VI. EDIT FACTORS.

All numeric fields will contain valid numeric data.

TA-03-05-14 (D)

"For Instructional Purposes Only"

S255

RECORD LAYOUT #2 1 <i>For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.</i>		PAGE NO <u>1</u> OF <u>1</u> PAGES		
		DATE December 19xx		
SYSTEM ID Truck Supply System	RECORD LENGTH 80 Characters	PREPARED BY SP6 D.W. Gross		
<input type="checkbox"/> CARD <input checked="" type="checkbox"/> DISK <input type="checkbox"/> TAPE <input type="checkbox"/> OTHER		FILE ID SYS006		
1 record per block		REMARKS ORDERS FILE (Create Records)		
RELATIVE POSITION	IDENTIFICATION OF ELEMENT (FIELD)	ABBREVIATION	LENGTH/ CLASS	LOCATION
1 2 3	PART NUMBER Quantity On Hand Not Used		4N 4N 72AN	1-4 5-8 9-80
Appendix A		3		

IA-03-05-14 (D)

"For Instructional Purposes Only"

S255

RECORD LAYOUT

For use of this form, see AR 18-7; the proponent agency is
Office of the Assistant Vice Chief of Staff.

PAGE NO. 1 OF 1 PAGES

DATE

December 19xx

SYSTEM ID

Truck Supply System

RECORD LENGTH

80 Characters

PREPARED BY

SP6 D.W. Gross

☐ CARD☒ DISK☐ TAPE☐ OTHER

FILE ID

SYS006

1 record
per block

REMARKS

ORDERS FILE (Order Record)

RELATIVE POSITION	IDENTIFICATION OF ELEMENT (FIELD)	ABBREVIATION	LENGTH/ CLASS	LOCATION
1	PART NUMBER			
1a	CLASS NUMBER (either a 1,2, or 3)		1N	1
1b	Part Number		3N	2-4
2	CUSTOMER NAME		18AN	5-22
3	CUSTOMER ADDRESS		18AN	23-40
4	UNIT OF ISSUE		2AN	41-42
5	QUANTITY ORDERED		3N	43-45
6	PRICE PER ITEM		5N*	46-50
7	CUSTOMER CODE		1AN**	51 ***
8	NOT USED		29AN	52-80
	* DOLLARS AND CENTS FIELD			
	** CAN ONLY BE A BLANK OR A '?'			
	*** Will be a 'D' for deletion transactions			
Appendix A		4		

FORM LAYOUT #1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

FOR INSTRUCTIONAL PURPOSES ONLY

5

3-221
COBOL ISAM FILE TEXT

FORM LAYOUT #2

402

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

FOR INSTRUCTIONAL PURPOSES ONLY

6

3-221

453

452

1

IDENTIFICATION DIVISION.
 PROGRAM-ID. PE-7EXAMPLE.
 AUTHOR. SP6DW GROSS
 INSTALLATION. CSD, USAIA, FT HARRISON IN 46216.
 DATE-WRITTEN. MAR 1, 1978
 DATE-COMPILED. JUL 5, 1978
 SECURITY. NONE

REMARKS. *****
 * GENERAL FUNCTIONS. *
 * THIS PROGRAM ILLUSTRATES *
 * 1 CREATING AN ISAM FILE *
 * 2 RANDOMLY UPDATING AN ISAM FILE *
 * 3 READING AND LISTING THE UPDATED ISAM *
 * FILE SEQUENTIALLY. *
 * *
 * SPECIFIC FUNCTIONS. *
 * *
 * THIS PROGRAM READS THE CREATE RECORDS *
 * AND BUILDS AN ISAM FILE. THE ORDER RECORDS *
 * FOR EACH DAY ARE THEN MATCHED AGAINST THE *
 * BALANCE FILE (ISAM FILE), AND THE QUANTITY *
 * ORDERED IS SUBTRACTED FROM THE BALANCE. *
 * FILES AUANTITY ON HAND. IF A NEGATIVE RE- *
 * SULT IS REACHED, A MESSAGE IS PRINTED TO *
 * THAT EFFECT ON THE TRANSACTION REPORT. *
 * FINALLY THIS PROGRAM LISTS THE UPDATED *
 * BALANCE FILE. *
 * *****

ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-360-H40.
 OBJECT-COMPUTER. IBM-360-H40.
 SPECIAL-NAMES.
 COI IS TOP-OF-PAGE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
 SELECT IM-ORDER-FILE
 ASSIGN TO UT-S-SYS006.
 SELECT OR-PRINTED-REPORTS
 ASSIGN TO UT-S-SYS005.

 *
 * NOTICE THE ASSIGN CLAUSE FOR THE FOL-
 * LOWING SELECTS. ISAM FILES MUST BE ASSIGN-
 * ED TO DIRECT ACCESS DEVICES, THUS THE 'DA
 * -I'. ALSO YOU MUST SPECIFY WHICH FIELD IN
 * THE RECORDS IS THE KEY FIELD. FOR RANDOM
 * PROCESSING, THE CLAUSES ACCESS IS RANDOM,
 * AND NOMINAL KEY IS MUST BE SPECIFIED.
 *

00031
 00032
 00033
 00034
 00035
 00036
 00037
 00038
 00039
 00040
 00041
 00042
 00043
 00044
 00045
 00046
 00047
 00048
 00049
 00050
 00051
 00052
 00053
 00054
 00055
 00056
 00057
 00058
 00059
 00060
 00061

SELECT BM-BALANCE-FILE
 ASSIGN TO DA-I-SYSC08
 RECORD KEY IS OM01-PART-NUMBER.
 SELECT 8M-BALANCE-FILE-RANDOM
 ASSIGN TO DA-I-SYS008
 ACCESS IS RANDOM
 RECORD KEY IS 8M02-PART-NUMBER
 NOMINAL KEY IS WS-PART-NUMBER

00062	DATA DIVISION.	
00063	FILE SECTION.	
00064	FD IM-ORDER-FILE	
00065	BLOCK CONTAINS 1 RECORDS	
00066	RECORD CONTAINS 80 CHARACTERS	
00067	LABEL RECORD IS STANDARD	
00068	DATA RECORDS ARE IM01-CREATE-REC	
00069	IM02-ORDER-REC.	
00070		
00071	01 IM01-CREATE-REC.	PIC 9(4).
00072	05 IM01-PART-NUMBER	PIC S9(4).
00073	05 IM01-QUANTITY	PIC X(72).
00074	05 FILLER	
00075	01 IM02-ORDER-REC.	PIC 9(4).
00076	05 IM02-PART-NUMBER	PIC X(18).
00077	05 IM02-CUSTOMER-NAME	PIC X(18).
00078	05 IM02-CUSTOMER-ADDRESS	PIC XX.
00079	05 IM02-UNIT-OF-ISSUE	PIC 999.
00080	05 IM02-QUANTITY-ORDERED	PIC 999V99.
00081	05 IM02-PURCHASE-AMOUNT	PIC X.
00082	05 IM02-CUSTOMER-CODE	PIC X(29).
00083	05 FILLER	
00084	FD OR-PRINTED-REPORTS	
00085	BLOCK CONTAINS 1 RECORDS	
00086	RECORD CONTAINS 133 CHARACTERS	
00087	LABEL RECORD IS STANDARD	
00088	DATA RECORD IS OR01-PRINT-LINE.	PIC X(133).
00089	01 OR01-PRINT-LINE	
00090	FD BM-BALANCE-FILE	
00091	BLOCK CONTAINS 1 RECORDS	
00092	RECORD CONTAINS 11 CHARACTERS	
00093	LABEL RECORD IS STANDARD	
00094	DATA RECORD IS BM01-BLAANCE-REC.	PIC X.
00095	01 BM01-BALANCE-REC.	PIC 9(4).
00096	05 BM01-DELETE-KEY	PIC XX.
00097	05 BM01-PART-NUMBER	PIC S9(4).
00098	05 FILLER	
00099	05 BM01-QUANTITY-ON-HAND	
00100	FD BM-BALANCE-FILE-RANDOM	
00101	BLOCK CONTAINS 1 RECORDS	
00102	RECORD CONTAINS 11 CHARACTERS	
00103	LABEL RECORD IS STANDARD	
00104	DATA RECORD IS BM02-BALANCE-REC-R.	PIC X.
00105	01 BM02-BALANCE-REC-R.	PIC 9(4).
00106	05 BM02-DELETE-KEY	PIC XX.
00107	05 BM02-PART-NUMBER	PIC S9(4).
00108	05 FILLER	
	05 BM02-QUANTITY-ON-HAND	

Appendix

COBOL ISAM FILE TEXT

```

00109 WORKING-STORAGE SECTION.
00110 77 WS-BEGIN-POINT PIC X(27)
00111 VALUE 'WORKING STORAGE BEGINS HERE'.
00112 77 WS-EOF-SWITCH PIC XXX
00113 VALUE 'OFF'.
00114 77 WS-ERROR-SW PIC XXX
00115 VALUE 'OFF'.
00116 77 WS-LINE-COUNTER PIC 99
00117 VALUE 26.
00118 77 WS-PART-NUMBER PIC 9(4)
00119 VALUE 0.
00120 01 WS-HEADING-LINE-1.
00121 05 FILLER PIC X(13)
00122 VALUE SPACES.
00123 05 WS-HL1-TITLE PIC X(18).
00124 05 FILLER PIC X(102)
00125 VALUE SPACES.
00126 01 WS-HEADING-LINE-2.
00127 05 FILLER PIC XX
00128 VALUE SPACES.
00129 05 FILLER PIC X(7)
00130 VALUE 'PART NO'.
00131 05 FILLER PIC X(4)
00132 VALUE SPACES.
00133 05 FILLER PIC X(8)
00134 VALUE 'QUANTITY'.
00135 05 FILLER PIC X(5)
00136 VALUE SPACES.
00137 05 WS-HL2-MESSAGE PIC X(7)
00138 VALUE 'MESSAGE'.
00139 05 FILLER PIC X(100)
00140 VALUE SPACES.
00141 01 WS-DETAIL-LINE.
00142 05 FILLER PIC X(4)
00143 VALUE SPACES.
00144 05 WS-DL-PART-NUMBER PIC 9(4).
00145 05 FILLER PIC X(7)
00146 VALUE SPACES.
00147 05 WS-DL-QUANTITY PIC ----9.
00148 05 FILLER PIC X(5)
00149 VALUE SPACES.
00150 05 WS-DL-MESSAGE PIC X(106).
00151 01 WS-END-POINT PIC X(25)
00152 VALUE 'WORKING STORAGE ENDS HERE'.

```

```

00153      PROCEDURE DIVISION.
00154      0000-DRIVER.
00155      *****
00156      *
00157      *           THE DRIVER PORTION OF THIS PROGRAM
00158      * DIRECTS THE EXECUTION THROUGH THE THREE
00159      * PARTS OF THIS PROGRAM.
00160      *
00161      *****
00162      OPEN INPUT IM-ORDER-FILE
00163      OUTPUT BM-BALANCE-FILE.
00164      PERFORM 0100-CREATE-ROUTINE THRU 0100-EXIT
00165      UNTIL WS-EOF-SWITCH EQUAL 'ON'.
00166      IF WS-ERROR-SW EQUAL 'ON'
00167      GO TO 0000-EXIT.
00168      CLOSE BM-BALANCE-FILE.
00169      *****
00170      *
00171      *           THE FOLLOWING OPEN STATEMENT ILLUSTRATES
00172      * THE 'I-O' OPTION. THIS MEANS THAT THE
00173      * FILE WILL BE OPENED AS BOTH INPUT AND OUTPUT.
00174      *
00175      *****
00176      OPEN I-O BM-BALANCE-FILE-RANDOM
00177      OUTPUT OR-PRINTED-REPORTS.
00178      MOVE 'OFF' TO WS-EOF-SWITCH.
00179      MOVE 'TRANSACTION REPORT' TO WS-HL1-TITLE.
00180      PERFORM 0200-UPDATE-ROUTINE THRU 0200-EXIT
00181      UNTIL WS-EOF-SWITCH EQUAL 'ON'.
00182      CLOSE IM-ORDER-FILE
00183      BM-BALANCE-FILE-RANDOM.
00184      OPEN INPUT BM-BALANCE-FILE
00185      MOVE 'OFF' TO WS-EOF-SWITCH.
00186      MOVE 26 TO WS-LINE-COUNTER.
00187      MOVE SPACES TO WS-OL-MESSAGE.
00188      MOVE SPACES TO WS-HL2-MESSAGE.
00189      MOVE 'ISAM FILE LISTING' TO WS-HL1-TITLE.
00190      PERFORM 0500-LIST-ROUTINE THRU 0500-EXIT
00191      UNTIL WS-EOF-SWITCH EQUAL 'ON'.
00192      CLOSE OR-PRINTED-REPORTS
00193      BM-BALANCE-FILE.
00194      0000-EXIT.
00195      STOP RUN.

```

```

00196 *****
00197 *
00198 *           THE CREATE ROUTINE READS EACH INPUT *
00199 * CREATE RECORD, MOVES THE FIELDS TO THE *
00200 * ISAM BALANCE RECORD, AND WRITES EACH REC- *
00201 * ORD. NOTICE THE WRITE STATEMENT. WHEN *
00202 * WRITING TO AN ISAM FILE THE INVALID KEY *
00203 * CLAUSE IS USED. IN THIS CASE IF THERE IS *
00204 * AN INVALID KEY IT MEANS THERE IS NO MORE *
00205 * SPACE LEFT ON THE DISK. *
00206 *
00207 *****
00208 0100-CREATE-ROUTINE.
00209 READ IM-ORDER-FILE
00210 AT END
00211 MOVE 'ON' TO WS-EOF-SWITCH
00212 GO TO 0100-EXIT.
00213 IF IM01-CREATE-REC EQUAL SPACES
00214 THEN MOVE 'ON' TO WS-EOF-SWITCH
00215 GO TO 0100-EXIT.
00216 MOVE LOW-VALUES TO BM01-DELETE-KEY.
00217 MOVE IM01-PART-NUMBER TO BM01-PART-NUMBER.
00218 MOVE IM01-QUANTITY TO BM01-QUANTITY-ON-HAND.
00219 WRITE BM01-BALANCE-REC
00220 INVALID KEY
00221 DISPLAY 'FULL DISK OR TRANS OUT OF SEQ',
00222 IM01-PART-NUMBER
00223 CLOSE IM-ORDER-FILE
00224 BM-BALANCE-FILE
00225 MOVE 'ON' TO WS-EOF-SWITCH
00226 MOVE 'ON' TO WS-ERROR-SW.
00227 0100-EXIT.
00228 EXIT.
00229 *****
00230 *
00231 *           THE FUNCTION OF THE UPDATE ROUTINE IS *
00232 * TO READ EACH ORDER RECORD, READ THE ISAM *
00233 * FILE RANDOMLY FINDING THE MATCHING RECORD *
00234 * AND ADJUST THE QUANTITY ON HAND. THEN IN- *
00235 * VALID KEY CLAUSE IS USED WITH THE READ *
00236 * STATEMENT, WHEN THE MATCHING RECORD IS NOT *
00237 * THERE, THIS CONDITION IS INVOKED. *
00238 * AFTER THE QUANTITY ON HAND IS ADJUSTED, *
00239 * THE ISAM RECORD IS REPLACED IN ITS PLACE *
00240 * ON THE FILE. THE REWRITE STATEMENT IS USED *
00241 * FOR THIS. *
00242 *
00243 *****
00244 0200-UPDATE-ROUTINE.
00245 READ IM-ORDER-FILE
00246 AT END
00247 MOVE 'ON' TO WS-EOF-SWITCH
00248 GO TO 0200-EXIT.
00249 MOVE IM02-PART-NUMBER TO WS-PART-NUMBER.
00250 READ BM-BALANCE-FILE-RANDOM
00251 INVALID KEY
00252 MOVE IM02-PART-NUMBER TO WS-OL-PART-NUMBER

```

```

00253      MOVE IM02-QUANTITY-ORDERED TO WS-DL-QUANTITY
00254      MOVE 'UNMATCHED PART NUMBER' TO WS-DL-MESSAGE
00255      PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT
00256      GO TO 0200-EXIT.
00257      IF BM02-DELETE-KEY EQUAL HIGH-VALUES
00258      THEN MOVE 'TRANSACTION FOR DELETED RECORD-NOT PROCESSED'
00259      TO WS-DL-MESSAGE
00260      MOVE IM02-PART-NUMBER TO WS-DL-PART-NUMBER
00261      MOVE IM02-QUANTITY-ORDERED TO WS-DL-QUANTITY
00262      PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT
00263      GO TO 0200-EXIT.
00264      IF IM02-CUSTOMER-CODE EQUAL 'D'
00265      THEN MOVE HIGH-VALUES TO BM02-DELETE-KEY
00266      MOVE 'DELETED RECORD' TO WS-DL-MESSAGE
00267      MOVE IM02-PART-NUMBER TO WS-DL-PART-NUMBER
00268      MOVE IM02-QUANTITY-ORDERED TO WS-DL-QUANTITY
00269      PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT
00270      REWRITE BM02-BALANCE-REC-R
00271      INVALID KEY
00272      DISPLAY 'RECORD NOT DELETED',
00273      IM02-PART-NUMBER
00274      GO TO 0200-EXIT.
00275      SUBTRACT IM02-QUANTITY-ORDERED FROM BM02-QUANTITY-ON-HAND.
00276      MOVE BM02-PART-NUMBER TO WS-DL-PART-NUMBER.
00277      MOVE BM02-QUANTITY-ON-HAND TO WS-DL-QUANTITY.
00278      IF BM02-QUANTITY-ON-HAND IS LESS THAN 0
00279      THEN MOVE 'NEGATIVE BALANCE ON HAND' TO WS-DL-MESSAGE
00280      ELSE
00281      MOVE 'PROCESSED' TO WS-DL-MESSAGE.
00282      PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT.
00283      REWRITE BM02-BALANCE-REC-R
00284      INVALID KEY
00285      DISPLAY BM02-PART-NUMBER, WS-PART-NUMBER.
00286      0200-EXIT.
00287      EXIT.
00288      0300-PRINT-ROUTINE.
00289      IF WS-LINE-COUNTER GREATER THAN 19
00290      THEN PERFORM 0400-HEADER-ROUTINE THRU 0400-EXIT.
00291      WRITE 0001-PRINT-LINE FROM WS-DETAIL-LINE
00292      AFTER ADVANCING 2 LINES.
00293      ADD 1 TO WS-LINE-COUNTER.
00294      MOVE SPACES TO WS-DETAIL-LINE.
00295      0300-EXIT.
00296      EXIT.
00297      0400-HEADER-ROUTINE.
00298      MOVE ZERO TO WS-LINE-COUNTER.
00299      WRITE 0001-PRINT-LINE FROM WS-HEADING-LINE-1
00300      AFTER ADVANCING TOP-OF-PAGE.
00301      WRITE 0001-PRINT-LINE FROM WS-HEADING-LINE-2
00302      AFTER ADVANCING 2 LINES.
00303      0400-EXIT.
00304      EXIT.
00305      0500-LIST-ROUTINE.
00306      READ BM-BALANCE-FILE
00307      AT END
00308      MOVE 'ON' TO WS-EOF-SWITCH
00309      GO TO 0500-EXIT.

```

410

8

00310 MOVE BM01-QUANTITY-ON-HAND TO WS-DL-QUANTITY.
00311 MOVE BM01-PART-NUMBER TO WS-DL-PART-NUMBER.
00312 PERFORM 0300-PRINT-ROUTINE THRU 0300-EXIT,
00313 0500-EXIT.
00314 EXIT.

461

TRANSACTION REPORT

PART NO	QUANTITY	MESSAGE
1941	-35	NEGATIVE BALANCE ON HAND
1268	993	PROCESSED
1767	2513	PROCESSED
1513	6749	PROCESSED
1052	5048	PROCESSED
1223	-9	NEGATIVE BALANCE ON HAND
1454	4256	PROCESSED
1689	9903	PROCESSED
1169	394	PROCESSED
1491	154	PROCESSED
1767	2507	PROCESSED
1268	981	PROCESSED
1513	6740	PROCESSED
1941	-285	NEGATIVE BALANCE ON HAND
1655	117	PROCESSED
1689	9901	PROCESSED
1655	53	PROCESSED
1454	4239	PROCESSED
1052	5047	PROCESSED
1513	6738	PROCESSED

412

TRANSACTION REPORT

PART NO	QUANTITY	MESSAGE
1689	9900	PROCESSED
1491	129	PROCESSED
1767	2499	PROCESSED
1454	4237	PROCESSED
1169	384	PROCESSED
1223	-29	NEGATIVE BALANCE ON HAND
2086	9742	PROCESSED
2019	9299	PROCESSED
2652	426	PROCESSED
2152	7341	PROCESSED
2925	1415	PROCESSED
2068	927	PROCESSED
2529	5827	PROCESSED
2019	9298	PROCESSED
2652	414	PROCESSED
2152	7332	PROCESSED
3109	999	PROCESSED
3327	1501	PROCESSED
3007	3579	PROCESSED
3318	7800	PROCESSED

463

TRANSACTION REPORT

PART NO	QUANTITY	MESSAGE
3765	4987	PROCESSED
3082	9921	PROCESSED
3007	3	DELETED RECORD
3007	3576	PROCESSED
3007	2	TRANSACTION FOR DELETED RECORD-NOT PROCESSED
3318	7799	PROCESSED
3327	1498	PROCESSED
3765	4986	PROCESSED
3109	959	PROCESSED
3082	9920	PROCESSED
3327	1490	PROCESSED
3765	4985	PROCESSED
9999	3	UNMATCHED PART NUMBER
3109	938	PROCESSED

414

ISAM FILE LISTING

PART NO	QUANTITY
1052	5047
1169	384
1223	-29
1268	981
1454	4237
1491	129
1513	6738
1655	53
1689	9900
1767	2499
1941	-285
2019	9298
2068	927
2090	9742
2152	7332
2529	5827
2652	414
2925	1415
3082	9920
3109	938

485

ISAM FILE LISTING

PART NU	QUANTITY
3318	7799
3327	1490
3765	4985

416

IA-01-02-10 (D)
IA-01-03-10 (D)

Assembler Language

Practical Exercises

**ALC Team, Software Division
Data Processing Department
United States Army Institute
of Administration**

DC's and DS's

ALC PE #1

- A. Write the statements necessary to define the heading shown on the attached form layout. (Form Layout 1)
- B. Write the statements necessary to define the detail line shown on the attached form layout. (Form Layout 1)
- C. Write the statements necessary to define an input area corresponding to the format shown on the attached card layout. (Card Layout 1)
- D. Write the statement to allocate enough space for the printer output area.

MULTIPLE-CARD LAYOUT																																													PAGE NO.																																		
INSTRUCTIONAL USE ONLY															For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.																																																																
CARD LAYOUT 1																														PREPARED BY															NO. OF PAGES																																		
SSN					NAME															RANK		DATE OF RANK					NOT USED																																																				
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

418



FORM LAYOUT

Form Layout 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

FOR INSTRUCTIONAL PURPOSES ONLY

3.221

One of the programmers in your shop has left you with a partially written ALC program. He has completed the logic portion of the program and all that remains to be written is the declarative section and the DTF's.

You are to define the following areas in storage:

1. A one byte field named SPACE initialized to blank.
2. A 132 byte field named OUTPUT, no initial value is needed. This field will be used to hold character information.
3. An 80 byte field called INPUT. This field should be defined with a 0 duplication factor.
4. A 79 byte field with no name attached. This field will hold character information, and no initial value is needed.
5. A one byte field named CODE. It will hold character data and needs no initial value.
6. A one byte field called DASH initialized to the value '-'.

The following DTF's are also needed:

1. A DTFPR. The I/O area is OUTPUT, the device address is SYSLST, the file name is PRINT, and the CNTRL macro will be used, the block size should be 132.
2. A DTFCD. The I/O area is INPUT, the device address is SYSIPT, the file name is CARD, and the end of file address is ENDJOB.

The above storage areas and DTF's should be coded in the order shown above. Once you have done the required coding, keypunch the cards and submit them for execution. The procedural part of the program will be supplied by the student job stream as will the data cards.

PERSONNEL LISTING

ALC PE #3

Assemble: A01

Execute: A03

You are required to write an ALC program to list all of the people in your unit.

INPUT: The input has the same format as the input for ALC PE #1.

The logical name for the card reader is SYSIPT.

OUTPUT: The heading and detail lines will have the same format as those for ALC PE #1. You may assume that all output will fit on one page. The logical name for the printer is SYSLST. Double space all output.

ASSEMBLE A01
EXECUTE A03

ALC PE 4

OFFICE LISTING

- I. Introduction: You are to write a program that will print a listing of the people assigned to the office. This listing will show each individual's name, phone number, home address, and marital status.
- II. Definitions: SCODE = The record selection code
MCODE = The marital status code
- III. Input/Output:
- A. Input: The input will be the personnel master file which is on cards (see attached card layout). Device address = SYSIPT.
- B. Output: The output will be a printed report on the 1403 line printer (see the attached form layout). Device address = SYSLST.
- IV. Assumptions:
- A. The listing will fit on one page
- B. The MCODE will be only an M, S, or D
- V. Methods:
- A. Provide a title and column headings as shown on the form layout
- B. List only those personnel that have an SCODE of zero. Use the CLC instruction to check the SCODE
- C. Using an MVI instruction, insert a hyphen between the third and fourth digit of the phone number on the printed report. This will require breaking the phone number down into two parts.
- D. Determine the individuals marital status using the CLI instruction. If the code is an "M", print "MARRIED" in the STATUS field; if the code is an "S", print "SINGLE"; and if the code is a "D", print "DIVORCED".
- VI. Edit Factors: None

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO.

1

SYSTEM

PREPARED BY

NO. OF PAGES

1

Card Layout 2

Phone
Number

Name

S
C
O
D
E

M
C
O
D
E

Address

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

423

FORM LAYOUT

424

FORM LAYOUT 2

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	OFFICE PERSONNEL LISTING												
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													
31													
32													
33													
34													
35													
36													
37													
38													
39													
40													
41													
42													
43													
44													
45													
46													
47													
48													
49													
50													
51													

FOR INSTRUCTIONAL PURPOSES ONLY

3.221



OVER 36

ALC PE 5

Assemble: A01
Execute: A03

You are to write an ALC program to produce a listing of all personnel in the battalion who have completed more than 35 months of service.

INPUT: The input is a personnel master file presently stored on punched cards. See Card Layout 3.

- OUTPUT:
- 1) The output is on a 132 print position printer.
 - 2) Print a heading at the top of each new page.
 - 3) Only 10 detail lines should be printed per page.
 - 4) Print a total line containing the total personnel with 36 or more months of service.
 - 5) Double space all lines except triple space between the total line and the last line.
 - 6) See Form Layout 3 for various line formats.

Assumptions: Not more than 999 persons will have more than 35 months of service. Since we can record only full months of service, the person must have 36 months of service or more to qualify.

- Methods:
- a) Read each card and print a detail line only for personnel with over 35 months of service.
 - b) Keep a count of the total number of personnel with more than 35 months of service. This total will be part of the final total line.
 - c) Do not edit the months of service figure. However, the total personnel figure should be edited with leading nonsignificant zeros suppressed except when the figure is 000 in which case the total figure should print as a single zero.

MULTIPLE CARD LAYOUT

PAGE NO.

INSTRUCTIONAL USE ONLY

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

SYSTEM

PREPARED BY

NO. OF PAGES

CARD LAYOUT 2

Social
Security
Number

Name

ETS

Date
of
Rank

Not Used

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Company

Rank

Number of Dependents

Battalion

Pay Grade

Months of Service

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DA FORM 3165
1 AUG 66

U.S. GOVERNMENT PRINTING OFFICE: 1970 - 381-047

426

FORM LAYOUT

FORM LAYOUT 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

FOR INSTRUCTIONAL PURPOSES ONLY

3-221

ASSEMBLE A01
EXECUTE A24

ALC PE 6

PAYROLL EXERCISE

- I. Introduction: You are to write a program that will calculate the net pay and withholding tax for each individual within the company. The report that the program generates will list each individual and the information pertaining to his pay. It will also show the total tax withheld for each department.
- II. Definitions: None
- III. Input/Output:
 - A. Input: The input is on cards which are sorted by department. The pay and tax rate fields have two implied decimal points (see attached card layout for input data format).
Device address = SYSIPT.
 - B. Output: The output will be on a 1403 line printer and will consist of column headings on each page, detail lines, and a total line for each department. Start each department on a new page (see attached form layout for output specifications).
Device address = SYSLST.
- IV. Assumptions:

Each department will fit on one page. The total tax figure will not exceed \$99999.99.
- V. Methods:
 - A. For each record calculate the withholding tax as follows:
 $\text{Withholding tax} = \text{Tax Rate} \times \text{Gross Pay}$
 - B. Calculate Net Pay as follows:
 $\text{Net Pay} = \text{Gross Pay} - \text{Withholding Tax}$
 - C. Keep track of the total taxes for each department. This will be part of the total line.
 - D. Zero suppress all numeric output fields. The Gross Pay, Withholding Tax, and Net Pay fields should have decimal points, commas, and fixed dollar signs as shown on the form layout.
- VI. Edit factors: None

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO.

1

SYSTEM

PREPARED BY

NO. OF PAGES

1

Card Layout 5

D
E
P
T

NAME

GROSS
PAY

T
R
A
T
E

TRATE = Tax Rate

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

3165

429

487

FORM LAYOUT

430

FORM LAYOUT 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
H	DEPT	NAME	GROSS PAY	TAX RATE	TAXES	NET PAY								
D	XX	X	\$2,221.00	15%	\$333.15	\$1,887.85								
D	XX	X	\$2,221.00	15%	\$333.15	\$1,887.85								
D	XX	X	\$2,221.00	15%	\$333.15	\$1,887.85								
D	XX	X	\$2,221.00	15%	\$333.15	\$1,887.85								
T	XX	X	\$2,221.00	15%	\$333.15	\$1,887.85								
					\$333.15									

FOR INSTRUCTIONAL PURPOSES ONLY



ALC PE 7

PARTS INVENTORY

- I. Introduction: The Fix or Repair Daily Corporation has decided to start selling by the case instead of breaking a case to sell individual items. They want you to write a program that will produce a report showing the number of full cases and the number of loose items for each stock number.
- II. Definitions: Loose Items = the remainder of the division of
Number of Items;
Number per Case
- III. Input/Output:
- A. Input: The input will be The Current Inventory File which is kept on cards (see card layout).
Device address = SYSIPT.
- B. Output: The output will be a printed report on the 1403 line printer (see the form layout for format and spacing).
Device address = SYSLST.
- IV. Assumptions: None.
- V. Methods:
- A. Provide main headings with a page number and column headings at the top of each page.
- B. Edit all numeric fields to ensure that they contain numeric information.
1. If they do not, bypass all calculations and print out the card with fields in the appropriate positions. Insure that the Loose Items and Number of Cases fields are blank. The word "ERROR" should also be printed on the same line.
 2. If they do, calculate the Number of Cases as follows:
$$\text{Number of Cases} = \frac{\text{Number of Items}}{\text{Number per Case}}$$
 3. Print a detail line consisting of Stock Number, Item Name, Number of Items, Number of Cases, and Number of Loose Items editing as shown on the form layout.
- C. Print only 20 double spaced detail lines per page.
- VI. Edit Factors: Edit all numeric input fields to ensure that they contain numeric information.

MULTIPLE-CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff

PAGE NO.

1

SYSTEM

CARD LAYOUT 6

PREPARED BY

NO. OF PAGES

1

STOCK
NUMBER

ITEM
NAME

NUMBER
OF
ITEMS

I
/
C

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

I/C = Items per case

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

432

492

FORM LAYOUT

FORM LAYOUT 6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	FOR THE REPUBLICAN PARTY CORPORATION													
2	PAGE 229													
3	STOCK #		ITEM NAME		# OF ITEMS		# OF CASES		# OF LOOSE ITEMS					
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

BLOOD TYPE

ALC PE 8

Assemble: A01

Execute: A05

You are required to determine the number of people with blood types A, B, AB, and O for a department in the Institute.

- INPUT: 1) The input consists of two files: A sequential disk master file and a card file.
- 2) The format of the sequential disk records is given in on disk Layout 7. The file has a blocking factor of 3.
- 3) The card format is shown on Card Layout 7.
- 4) The blood type field might look like the following for example:

record position 31

79

ABCAABBOOCCCCAAAABOOOOOAC...ABB

The blood type codes have the following meaning:

<u>Blood Code</u>	<u>Blood Type</u>
A	A
B	B
C	AB
O	O

Position 80 has a code. If it is anything other than the letter 'B', ignore that disk record.

- 5) The records are sorted by departments and one department may be on more than one record.
- 6) The card file contains one card with two numbers - a low key and a high key; these two numbers should be used to select the disk records to be processed. Only if the following condition exists should the disk record be processed:

low key \leq disk record position 1-10 \leq high key

If the condition does not exist, ignore that record.

- OUTPUT: 1) The output will be on a 132 print position printer and consist of six lines for each organization.
- 2) Each organization should be on a separate page with the format given on Form Layout 7.

- Assumptions:
- a) When the disk file was created, the name assigned to the file through Job Control was DISKIN. The device address is SYS006.
 - b) Ignore all illegal blood types found.
 - c) The totals can be expected to be at most 5 digits long.
 - d) Edit the totals by suppressing leading non-significant zeros.

FORM LAYOUT

form layout 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														
52														
53														
54														
55														
56														
57														
58														
59														
60														
61														
62														
63														
64														
65														
66														
67														
68														
69														
70														
71														
72														
73														
74														
75														
76														
77														
78														
79														
80														
81														
82														
83														
84														
85														
86														
87														
88														
89														
90														
91														
92														
93														
94														
95														
96														
97														
98														
99														
100														

ORGANIZATION NUMBER 9-9

ORGANIZATION NAME 9-9-9

TOTAL TYPE A 9-9

TOTAL TYPE B 9-9

TOTAL TYPE C 9-9

TOTAL TYPE D 9-9

ALC PE. 9

MASTER FILE EDIT

I. Introduction: You are required to write a program that will edit a Transaction File and a Master File to ensure that they meet certain requirements. These files, if there are no edit errors, will be used as input for another program.

II. Definitions: None.

III. Input/Output:

Input: The Transaction File will be on cards (see card layout for specifications). The Master File will be on sequential disk. There are three records per block (see disk layout for specifications).

Card is on SYSIPT, Disk is on SYS006, File name for Disk is MASTIN.

Output: A printed report will be produced to show the status of the different records (see form layout for specifications).

Device address = SYSLST.

IV. Assumptions: There will be one, and only one, transaction record per master record.

V. Methods:

1. Print a main heading with page number and column headings at the top of each page.
2. Print 15 double spaced detail lines per page.
3. Set up a one byte switch to be used when you edit the input.
4. Edit the classification field on the transaction card to find out whether it is blank, numeric, alphabetic, or alpha numeric. One of the first four bits of the switch in Step 3 above will be turned on to indicate what type of field it is.
5. Do the same thing for the classification field on the master record, except that one of the last four bits will be turned on to indicate what type of field it is.

6. The code on the transaction card will tell you what type this field should be.

CLASSIFICATION

<u>TYPE OF CODE</u>	<u>TRANSACTION</u>	<u>MASTER</u>
1	Blank	Blank
2	Anything	Numeric
3	Alphabetic	Not blank
4	Numeric	Numeric

7. If either or both of the fields are not the appropriate type, print the fields in the correct print positions along with the word "ERROR"
8. If both the fields are the appropriate type, print the fields in the correct print positions.

VI. Edit Factors: None

440

The following definitions of terms apply to ALC PE #4. These are further explained by the series of examples that follow the definitions.

1. BLANK - this means that all 10 positions of the field are blanks.
2. NON-BLANK - This means that at least 1 and maybe as many as all 10 positions of the field contain anything other than a blank.
3. NUMERIC - This means that all 10 positions of the field contain only the digits 0 thru 9.
4. ALPHABETIC - This means that all 10 positions of the field contains only the letters A thru Z, or blanks, however, all 10 positions may not be blanks.

EXAMPLES:

<u>DATA FIELD</u>	<u>TYPE</u>
ABCDEFGHIJ	ALPHABETIC, NON BLANK
12345678	NON BLANK
.(AXC6.*)/	NON BLANK
1234567890	NUMERIC, NON BLANK
BBBBBBBBBB	BLANK
BBB642BBB	NON BLANK
XXABCXX6XX	NON BLANK
2468013579	NUMERIC, NON BLANK
JOHNBD0EB	ALPHABETIC, NON BLANK

503

MULTIPLE-CARD LAYOUT
(AR 18-7)

PAGE NO.

SYSTEM

PREPARED BY

NO. OF PAGES

TRANSACTION CARD

CARD LAYOUT 8

1

CLASSIF-
ICATION

C
O
D
E

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

26

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1777

DISK LAYOUT

For use of this form, see AR 18-7;
the proponent agency is Office of
the Comptroller of the Army.

TAPE NO.

PREPARED BY

DATE

PAGE NO.

REMARKS

MASTER FILE DISK LAYOUT 8

NO. OF
PAGES

NO., LENGTH, BLOCK

100 x 3

NOT USED	
1	50

NOT USED	CLASSIFICATION
51	99 01
	100

27

DA FORM 3166
1 AUG 86

244



FORM LAYOUT

FORM LAYOUT 8

TITLE	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	MASTER ENIT REPORT													
2	PAGE 2287													
3														
4														
5	TRANSACTION													
6	CORE													
7	MASTER													
8	XXXXXXXXXX													
9	X													
10	XXXXXXXXXX													
11	XXXXXXXXXX													
12	XXXXXXXXXX													
13	XXXXXXXXXX													
14	XXXXXXXXXX													
15	XXXXXXXXXX													
16	XXXXXXXXXX													
17	XXXXXXXXXX													
18	XXXXXXXXXX													
19	XXXXXXXXXX													
20	XXXXXXXXXX													
21	XXXXXXXXXX													
22	XXXXXXXXXX													
23	XXXXXXXXXX													
24	XXXXXXXXXX													
25	XXXXXXXXXX													
26	XXXXXXXXXX													
27	XXXXXXXXXX													
28	XXXXXXXXXX													
29	XXXXXXXXXX													
30	XXXXXXXXXX													
31	XXXXXXXXXX													
32	XXXXXXXXXX													
33	XXXXXXXXXX													
34	XXXXXXXXXX													
35	XXXXXXXXXX													
36	XXXXXXXXXX													
37	XXXXXXXXXX													
38	XXXXXXXXXX													
39	XXXXXXXXXX													
40	XXXXXXXXXX													
41	XXXXXXXXXX													
42	XXXXXXXXXX													
43	XXXXXXXXXX													
44	XXXXXXXXXX													
45	XXXXXXXXXX													
46	XXXXXXXXXX													
47	XXXXXXXXXX													
48	XXXXXXXXXX													
49	XXXXXXXXXX													
50	XXXXXXXXXX													
51	XXXXXXXXXX													

443

BASE PAY EXERCISE

I. Introduction: You are to write a program that will produce a one page report showing the current base pay for all enlisted personnel with less than 20 years service.

II. Definitions: None.

III. Input/Output:

Input: None:

Output: The output will be on a 1403 line printer (SYSLST), and will consist of a one page report as shown on the attached Form Layout.

IV. Assumptions: None.

V. Methods:

1. Print the column heading on the top of the page.
2. A macro called EMPAY has been catalogued on the system and it will generate the required nine pay tables, each table having eleven entries. Each entry of the tables will be a packed four byte field containing the old base pay. The label that you code on the macro will appear as the name of each table and will be followed by a number from one to nine which indicates which of the tables the label applies to. There are no operands required on the macro.
3. Each table will be in ascending order based on time in service.
4. Update each entry in all nine tables as follows:
$$\text{New Base Pay} = \text{Old Pay} + \text{Old Pay} \times 5.52\%$$
5. When the tables have been updated print out the detail lines as shown on the Form Layout.
6. Edit the pay fields with a floating dollar sign, and a decimal pnt.
7. Use at least one BCT, BXLE, or BXH in your program.

VI. Edit Factors: None.

FORM LAYOUT

FORM LAYOUT 9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2	UNDER 2		OVER 2	OVER 3	OVER 4	OVER 5	OVER 6	OVER 8	OVER 10	OVER 12	OVER 14	OVER 16	OVER 18	
3	E-1	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99	\$22229.99
4														
5	E-2	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
6														
7	E-3	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
8														
9	E-4	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
10														
11	E-5	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
12														
13	E-6	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
14														
15	E-7	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
16														
17	E-8	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
18														
19	E-9	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
20														
21	E-10	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
22														
23	E-11	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
24														
25	E-12	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
26														
27	E-13	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
28														
29	E-14	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
30														
31	E-15	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
32														
33	E-16	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
34														
35	E-17	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
36														
37	E-18	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
38														
39	E-19	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
40														
41	E-20	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
42														
43	E-21	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
44														
45	E-22	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
46														
47	E-23	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
48														
49	E-24	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____
50														
51	E-25	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____

EDIT ALL DOLLAR AMOUNTS WITH A FLOATING \$

445

BINARY MATH EXERCISE

- I. Introduction: You have been interviewed for an ALC programming position by the Southeastern Box Company. They are interested in hiring you but prior to making their decision, they want you to submit a sample program. This company uses binary math for all calculations, so your sample program will demonstrate your ability in the use of binary math instructions. The firm also requires detailed source documentation on all ALC programs so this will be an additional requirement on this sample.
- II. Definitions: None.
- III. Input/Output:
- A. Input: The input will be on cards (see attached card layout).
Device address = SYSIPT.
- B. Output: The output will be on the 1403 line printer (see attached form layout). **Device address = SYSLST.**
- IV. Assumptions: None.
- V. Methods:
- A. Print a title and column headings at the top of each page.
- B. Using a line counter, print only 20 double spaced detail lines per page.
- C. Initialize a field called TOTAL at zero.
- D. The indicator field on the input, when valid, will contain a one, two, or space. When it contains a one, bypass all ADD instructions until you encounter a two or a space in the indicator field.
- E. The processing code from the input will determine what calculations should be performed on the number read from the card. If the processing code is a --
- one - add the number to TOTAL;
- two - Subtract the number from TOTAL
- three - Divide the number into TOTAL keeping the quotient as the new TOTAL
- four - Multiply the number times TOTAL
- five - Add the number to TOTAL, then Multiply the new TOTAL by the number

six - Divide the number into TOTAL, then subtract the number from the new TOTAL

*** Any processing code that is larger than 6 is invalid.

- F. Print out each card with the contents of TOTAL out to the right of the card.
- G. If any errors are found on the card, bypass all calculations and print out the card, leaving the TOTAL field blank.
- H. If the TOTAL is negative, edit a minus sign on the right side of the TOTAL field on the output.

VI. Edit Factors: Edit the card to insure that all data fields are numeric, except the indicator field which may be a space.

MULTIPLE CARD LAYOUT

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

PAGE NO.

1

NO. OF PAGES

1

CARD LAYOUT 10

NUMBER

P

I

NOT USED

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

P = Processing Code

I = Indicator

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DA FORM 3165
1 AUG 66

U.S. GOVERNMENT PRINTING OFFICE: 1970 - 301-807

8777



516

FORM LAYOUT

FORM LAYOUT 16

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	BINARY INSTRUCTION EXERCISE													
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

FOR INSTRUCTIONAL PURPOSES ONLY

3. 221

6777

INTEGER ARITHMETIC

ALC PE 12

Assemble: A01
Execute: A12

You are to write an ALC program that will determine which numbers in a given interval are perfect squares, and which numbers are prime. (Prime numbers are those numbers that are only divisible by 1 and themselves).

INPUT: The program will read one card. This card will contain the upper and lower bounds of the interval. The format of the card is shown on Card Layout 11.

OUTPUT: The output should start at the top of a new page. Detail lines should follow the appropriate headings.

Each detail line will consist of 16 8 character fields. Each field will contain one number, right justified. See Form Layout 11.

Useful Information:

1. If a number is a perfect square it is not a prime.
2. If a number other than 2 is even it cannot be a prime.
3. If checking a number N to see if it is prime, it is only necessary to check for divisibility by numbers $\leq \sqrt{N}$.

MULTIPLE-CARD LAYOUT

PAGE NO.

INSTRUCTIONAL USE ONLY

For use of this form, see AR 18-7; the proponent agency is Office of the Assistant Vice Chief of Staff.

SYSTEM

PREPARED BY

NO. OF PAGES

CARD LAYOUT II

Lower Bound Upper Bound

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

36

DA FORM 3165

FORM LAYOUT

452

FORM LAYOUT

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

THE PRIME NUMBERS BETWEEN 9---9 AND 9---9

THE PERFECT SQUARES BETWEEN 9---9 AND 9---9

FOR INSTRUCTIONAL PURPOSES ONLY

3.221



SORT

I. Introduction: The current personnel roster is maintained on disk as a sequential file, but due to additions the file is no longer in the proper order. You are to write an ALC program that will read the disk file, sort the records, and rewrite the records in ascending SSN order.

II. Definitions: None.

III. Input/Output:

A. Input: Your input is on sequential disk in the format shown on the attached disk layout. The filename for your input file is DISKINP and it is at device address SYS005.

B. Output: Your output will be on sequential disk and will be in the same format as it was on the input. The filename for output is DISKOPT and the device address is SYS006.

IV. Assumptions: There will be no more than 30 records on the file.

V. Methods: You may use any strategy you want to sort the records, but you must do your sort in core.

One method of sorting is the 'Bubble Sort'. To use this method first read all records into memory. Then take the first record and compare it to all subsequent records. If you find a SSN lower than the SSN you are using switch the two records and use the new SSN to continue the compares. At the end of the first pass you will have the record with the lowest SSN in the first position. Repeat the process beginning with the second record, then the 3rd and so forth until you have repeated it for all but the last record. At this point all of the records will be in the correct order.

EXAMPLE:

9, 6, 2, 3, 1, 4, 5, 7, 8

We would compare '9' to '6'. Since '6' is less than '9' we switch the '6' and the '9' and continue comparing with the '6'. Continuing with the '6' we find that '2' is less than '6' so once again we switch and continue this time with the '2'. At the end of the first pass the string would look like this: 1, 9, 6, 3, 2, 4, 5, 7, 8.

For the second pass we start with the second record which in this case is the '9' again. We compare it to the '6' (3rd record) switch and continue with the '6' and so forth. At the end of the second pass it would look like this: 1, 2, 9, 6, 3, 4, 5, 7, 8.

VI. Edit Factors: None.

VII. Special Note: The output disk will be dumped for you so that you may check your results. You need do nothing to cause this to happen.

DISK LAYOUT For use of this form, see AR 18-7; the proponent agency is Office of the Comptroller of the Army.	TAPE NO.	PREPARED BY	DATE	PAGE NO. 1
	REMARKS DISK LAYOUT 12			NO. OF PAGES 1

454

NO., LENGTH, BLOCK

ORG	SSN	NAME	PG*	RANK	ETS	DATE OF RANK	MON SER
1 3 4	1 1 2 3		3 3 3 3 3 3 0 1 2 3 5 6			4 4 1 2	4 4 5 7 8 0

-----MISCELLANEOUS INFORMATION -----							
5 1			8 0				

* PAY GRADE							



FILL REPORT GENERATOR

ALC PE 14

Assemble: A01
Execute: A13

You are to write an ALC program that will generate a bar graph showing the percentage fill for all Federal Stock Numbers.

INPUT: Your program will read the fill data from an unlabeled tape. The tape may contain more than one entry for each FSN. The format of the tape is shown on Tape Layout 13.

The filename for the tape is TAPEIN and the Device address is SYS007.

OUTPUT: The output will consist of a heading at the top of each page and a number of detail lines (one for each FSN). The detail lines will contain three fields; the FSN, the percentage fill, and a row of asterisks representing the percentage fill (one asterisk per percent). See Form Layout 13.

The device address is SYSLST.

Notes: You will have to watch for zero fill items and items which have not been ordered during the period. There is also the possibility that there may be errors on the tape and that the quantity filled will be greater than the quantity ordered. Print zero fill items as part of the graph. List the zero order items on a separate page, one per line under an appropriate heading (See Form Layout 13b). Also on this final page should be one line with the phrase 'LOWEST FILL PERCENTAGE WAS FOR', followed by the FSN of the item with the lowest fill percentage and

456

the percentage. If there are two with the same percentage the one with the largest quantity ordered should be listed. If there are more of an item listed as filled than were ordered, list the FSN as part of the graph but instead of putting in a percentage put in the word 'ERROR'.

523

FORM LAYOUT

FORM LAYOUT 13

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														

HEADING

FIELD REPORT

PAGE 99

DETAIL

FSN

REFERENCES

TAPE LAYOUT

For use of this form, see AR 18-7;
the proponent agency is Office of
the Comptroller of the Army.

TAPE NO.

PREPARED BY

DATE

PAGE NO.

1

REMARKS

Tape Layout 13

NO. OF
PAGES
1

NO., LENGTH, BLOCK

21 X 1

FSN	QTY Ordered	QTY Filled	
9	99	99	9

DA FORM 3166

1 AUG 66

USAFPP 294-65 10/74

"FOR INSTRUCTIONAL PURPOSES ONLY"

D-500

532

FORM LAYOUT

136

0 1 2 3 4 5 6 7 8 9 10 11 12 13

FULL REPORT

PAGE 99

X - FEW - X

X - FEW - X

X - FEW - X

LOWEST FULL PERCENTAGE WAS FOR X - FEW - X

FOR INSTRUCTIONAL PURPOSES ONLY

EDIT PROBLEM

ALC PE 15

Assemble: A01

Execute: A14

You are to write an ALC program to perform edit checks on cards that will be input to a requisition processing run.

INPUT: The format is as shown on Card Layout 14,

The fields should be edited as follows:

The FSN should be numeric (0-9).

The nomenclature may contain alphanumeric information (0-9 or A-Z) or blanks or dashes.

The unit of measure should be alphabetic (A-Z)

The quantity should be numeric (0-9).

The activity code should contain alphabetic information in the first two positions and numeric information in the last four.

OUTPUT: The heading is as shown on Form Layout 14,

The detail line should consist of the card image and the name of the column in error. Use FSN for federal stock number and NONE if no errors are found on the card.

The logical name for the printer is SYSLST, double space all output

MULTIPLE-CARD LAYOUT

INSTRUCTIONAL USE ONLY

For use of this form, see AR 18-7, the proponent agency is Office of the Assistant Vice Chief of Staff

PAGE NO.

SYSTEM

PREPARED BY

NO OF PAGES

CARD LAYOUT

14

Federal Stock
Number

Nomenclature

UM

Quan

Actvty
Code

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

FORM LAYOUT

462

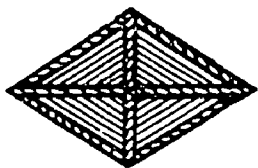
FORM LAYOUT

	0					1					2					3					4					5					6					7					8					9					10					11					12					13				
1																																																																						
2																																																																						
3																																																																						
4																																																																						
5																																																																						
6																																																																						
7																																																																						
8																																																																						
9																																																																						
10																																																																						
11																																																																						
12																																																																						
13																																																																						
14																																																																						
15																																																																						
16																																																																						
17																																																																						
18																																																																						
19																																																																						
20																																																																						
21																																																																						
22																																																																						
23																																																																						
24																																																																						
25																																																																						
26																																																																						
27																																																																						
28																																																																						
29																																																																						
30																																																																						
31																																																																						
32																																																																						
33																																																																						
34																																																																						
35																																																																						
36																																																																						
37																																																																						
38																																																																						
39																																																																						
40																																																																						
41																																																																						
42																																																																						
43																																																																						
44																																																																						
45																																																																						
46																																																																						
47																																																																						
48																																																																						
49																																																																						
50																																																																						
51																																																																						

FOR INSTRUCTIONAL PURPOSES ONLY

3.221





UNITED STATES ARMY INSTITUTE OF ADMINISTRATION

MAY 1978

PERFORMANCE TEST

DEBUGGING A COBOL PROGRAM USING ALC



Name _____

Student Number _____ Class Number _____

INSTRUCTIONS:

1. INDIVIDUAL WORK IS MANDATORY. Appropriate disciplinary action will be taken against students who give or receive unauthorized help.
2. Special instructions are listed on page 1 of this booklet.
3. Insure that your class and student number are on the individual grade sheet and/or punchee cards.

DEBUGGING COBOL PROGRAMS

Programs which do not go to a normal end of job are canceled by the operating system, and are considered to have abnormally terminated (ABEND). When this happens, page 1 of the program listing will contain, among other items, a system completion code (See Incl 1). The completion code is an indication of what caused the program to ABEND. Following is a list of the most commonly encountered system completion codes, a description of each, and an indication of the probably cause of the ABEND. Those causes marked with an asterisk (*) should be checked first.

COMPLETION CODE - 001.a. Error Description.

Input/Output error.

b. Probable Causes.

(*) Wrong length record.

(*) Wrong length physical block.

No end of file marker.

(*) Attempt to read record after end-of-file condition found.

Physical damage to recording medium.

Device malfunction.

(*) Attempt to read from or write to a file which—

(1) Is closed.

(2) Was not opened.

COMPLETION CODE - 013.a. Error Description.

Detection of conflicting or unsupported parameters during OPEN processing.

b. Probable Cause.

(*) Check for incorrect external name in a SELECT statement.

COMPLETION CODE - 031.a. Error Description.

An input/output error was detected when processing under the indexed sequential access method (ISAM).

b. Probable Causes.

Physical damage to recording medium or device.

(*) Out of sequence key when loading an ISAM data set.

(*) Wrong length record or block.

COMPLETION CODE - 03B.

a. Error Description.

The error occurred during OPEN processing for an indexed sequential file (ISAM).

b. Probable Causes.

(*) The file had not been created.

The file had not been closed after creation.

COMPLETION CODE - 03D.

a. Error Description.

The error occurred during OPEN processing for an indexed sequential file (ISAM).

b. Probable Causes.

Indexed sequential organization not specified.

(*) Check for omission of "ACCESS IS RANDOM" clause in select statement for ISAM file.

COMPLETION CODE - OC1.

a. Error Description.

The operation code detected is not valid.

b. Probable Causes.

A branch to a data area; fetching of data as an operation code for this instruction.

(*) A missing or misspelled external name on SELECT statement.

(*) A file had not been opened when an input/output instruction was issued for it.

A file had been closed when an input/output instruction was issued for it (this may also cause an OC5 termination).

466
COMPLETION CODE - OC5.

a. Error Description.

An address is specified that is outside of the available storage of the particular computing system.

b. Probable Causes.

Invalid data address.

(*) Indexing (subscripting) outside the program's assigned limits.

(*) Uninitialized index (or subscript). This may also cause a data exception (OC7).

An input/output instruction triggered termination because OPEN was unable to complete its task.

A missing or misspelled external name in SELECT statement.

An attempt to CLOSE a file a second time.

(*) COBOL: An improper exit from a procedure being operated on by a PERFORM statement.

COBOL: A sort operation is being attempted with an incorrect cataloged procedure.

(*) COBOL: An attempt to reference an input/output area before READ or OPEN statement has been issued for the file.

COMPLETION CODE - OC7.

a. Error Description.

Data in a field was of incorrect format for the instruction attempting to process it.

b. Probable Causes.

(*) A data field was not initialized, e.g., blanks were read into a field designed to be processed with packed decimal instructions.

A packed decimal field had an incorrect sign field.

Uninitialized index or subscript. This may also cause an addressing (OC5) or protection (OC4) completion code.

Fields in decimal arithmetic overlap incorrectly.

The decimal multiplicand has too many high-order significant digits.

(*) The index (or subscript) value was incorrect and invalid data was referenced. This could also cause an addressing (OC5) or protection (OC4) completion code.

COBOL: Data was moved from the DISPLAY field to the COMPUTATIONAL or COMPUTATIONAL-3 field at group level. No conversion was performed; invalid data for COMPUTATIONAL-3 (packed decimal) results.

COBOL: The figurative constants ZERO or LOW-VALUE was moved to a group level numeric field..

COBOL: Omission of USAGE clause or inclusion of an erroneous USAGE clause.

FOR USAIA USE: (See Incl 1)

- a. Extract Program Interruption Address (PIA) from page 1 of program listing.
- b. Determine entry point (EPA) for program from page 1 of program listing. It will be either—

(1) 01C020.

(2) 034020.

c. Subtract EPA from PIA. The result is a displacement. Remember, all addresses are in hexadecimal as is the displacement. Now go to the page in the listing which has the heading 'CONDENSED LISTING'. Each verb used in the source program is listed and has associated with it, a statement number and a displacement. To find the statement which caused the program to ABEND, locate the displacement on the condensed listing which most nearly matches and is less than the displacement you just calculated. The corresponding statement number is the one which caused the problem. Now, go back to the probable cause paragraph and correct problem.

COMPLETION CODE - OC9 OR OCB.

- a. Error Description.

Divide exception.

- b. Probable Cause.

Attempting to divide by zero.

COMPLETION CODE - D37.

- a. Error Description.

Space allocated for a data set on a direct access device was exceeded.

- b. Probable Cause.

The program may be in a loop and is writing an infinite number of records to an output file.

468

COMPLETION CODE - 222.

a. Error Description.

Operator canceled the job.

b. Probable Causes.

The job may have been canceled because it appeared to be in a loop or because it was waiting for resources that were not immediately available. Perhaps the job was canceled to correct a system interlock condition. There are many reasons why an operator might cancel a job. There may be nothing wrong with your program.

COMPLETION CODE - 322.

a. Error Description.

Execution of a program took longer than the time allocated.

b. Probable Cause.

Program is in a loop.

546


```
//TCELKES JOB (COB,1000,5,2,,,N.),
// 'GENERAL JOB',CLASS=B,MSGLEVEL=(1,5)
IEF142I - STEP WAS EXECUTED - CCND
IEF373I STEP /CCB / START 78
IEF374I STEP /CCB / STCP
IEF142I - STEP WAS EXECUTED -
IEF373I STEP /LKED / START
IEF374I STEP /LKED / STCP
CONTROL BYTE=CO TCB FLAGS=A
COMPLETION CODE - SYSTEM=OC7 USER=0000
PROGRAM INTERRUPTION (DATA) AT LOCATION 03517C
REGISTER SET 1
```

SYSTEM
COMPLETION CODE

MIN 51.155SEC MAIN 82K LCS OK

MIN 05.265SEC MAIN 84K LCS OK

INTERRUPT
ADDRESS

```
GPR 0-7 00034EEA 00035176 00034EC4 0003525C 000343C8
GPR 8-15 0004F368 0004F220 0004EC5 0003432 00034808 00034578 4034CA
INSTRUCTION IMAGE F27101006 09F9110106C150
FPR 0-4 00000000 00000000 00000000 00000000 00000000 00000000 00000000
ACTIVE RB LIST
```

```
PROGRAM ID= 801C RB TYPE=00 ENTRY POINT=0062F8
RESUME PSW SM= K= AMWP=4 IC=0007 IL+CC=5 FM=
PROGRAM ID=COB RB TYPE=00 ENTRY POINT=034020
RESUME PSW SM=FF K=3 AMWP=5 IC=000D IL+CC=0 FM=
```

ENTRY POINT
ADDRESS

```
IEF242I ALLCC. FOR TCCELKES GO COB AT ABEND
IEF237I 143 ALLCCATED TO PGM=*.CO
IEF237I 211 ALLCCATED TO SYSCL1
IEF237I 212 ALLCCATED TO SYS005
IEF237I 141 ALLCCATED TO SYS006
IEF237I 142 ALLCCATED TO SYS007
IEF237I 143 ALLCCATED TO SYS008
IEF285I SYS78082.T132344.RFO00.TCELKES.COSET
IEF285I VOL SER NOS= MFTWK3.
IEF285I USAIA.SOFTWARE.DATASET.012
IEF285I VOL SER NOS= MFTWK1.
IEF285I USAIA.SOFTWARE.DATASET.013
IEF285I VOL SER NOS= MFTWK1.
IEF285I SYS78082.T132344.RFO00.TCELKES.TEMP
IEF285I VOL SER NOS= MFTWK3.
IEF373I STEP /GO / START 78082.1334
IEF374I STEP /GO / STCP 78082.1335 CPL
IEF285I SYS78082.T132344.RFO00.TCELKES.COSET
IEF285I VOL SER NOS= MFTWK3.
IEF375I JOB /TCCELKES / START 78082.1329
IEF376I JOB /TCCELKES / STOP 78082.1335 CPL
```

MIN 03.04SEC MAIN 16K LOS OK
DELETED

MIN 59.46SEC

SUBTRACT THE ENTRY POINT ADDRESS
FROM THE INTERRUPT ADDRESS:

03517C
- 034020

00115C

CONDENSED LISTING

```

195  OPEN      000902
201  OPEN      000E4E
205  MOVE      000E8A0
209  CLOSE     0008DA
215  MOVE      000068
219  IF        000094
223  MOVE      000006
227  GO        000CEA
230  GO        000D1A
233  GO        000D4A
236  GO        000D7A
239  GO        000DAA
244  READ      000DD4
248  PERFORM   000E00
253  WRITE     000E2A
258  IF        000E74
261  ELSE      000EA4
263  MOVE      000EAE
267  EXIT      000FDA
272  IF        000F22
275  MOVE      000F54
278  IF        000F76
281  MOVE      000F90
284  IF        000F06
287  MOVE

```

USING THE CONDENSED LISTING,
DETERMINE THE COBOL STATEMENT
CAUSING THE INTERRUPT.

```

290  MOVE      001002
296  MOVE      0010EE
300  MOVE      001122
304  IF        001156
307  MOVE      001108
310  MOVE      001182
315  MOVE      0011FE
322  IF        00120A
325  MOVE      001268
328  WRITE     0012CA
331  MOVE
334  WRITE
337  WRITE

```

```

197  PERFORM   000A40
202  MOVE      000EE6
206  MOVE      000EAA
212  STOP      000C56
216  READ      000C6E
220  MOVE      000CA0
224  IF        000CCC
228  IF        000CF0
231  IF        000D20
234  IF        000D50
237  IF        000D60
240  PERFORM   000D60
246  MOVE      000DF4
250  EXIT      000E1E
255  EXIT      000E68
259  PERFORM   000E80
261  MOVE      000EA4
264  PERFORM   000EB4
269  MOVE      000F10
273  MOVE      000F38
276  IF        000F5A
279  MOVE      000FEC
282  IF        000FA2
285  MOVE      000FE4
288  MOVE      00100E
292  EXIT      001074
297  MOVE      00108C
302  MOVE      0010B6
305  IF        0010CC
308  MOVE      001100
311  PERFORM   001132
316  WRITE     001168
323  MOVE      0011CE
326  MOVE      0011EC
329  MOVE      0011FE
332  MOVE      001228
335  IF        0012EA
340  ADD        001306

```

STATISTICS SOURCE RECORDS = 342 DATA DIVISION STATEMENTS = 101
 OPTIONS IN EFFECT SIZE = E6 I6 BLF = 8192 EINECNT = 57 SFACET,
 OPTIONS IN EFFECT NCCMAP, NCPMAP, CLIST, SUPMAP, NCXREF, LOAD, NC
 OPTIONS IN EFFECT 2WB

00292

EXIT.

00293

0060-DELETE.

00294

MOVE 'CNN' TO WS-DM-READ-SW,

00295

WS-TRANS-READ-SW.

00296

MOVE IM01-EM-RCD TO OM 1-NEM-RCD.

00297

MOVE 'DELETION' TO WS-DL-STATUS.

00298

PERFORM 0080-WRITE THRU 0080-EXIT.

00299

0060-EXIT.

00300

EXIT.

00301

0070-ERROR.

00302

MOVE 'CNN' TO WS-TRANS-READ-SW.

00303

MOVE IT01-TRANS-RCD TO CM 1-NEM-RCD.

00304

MOVE 'BAD UPD' TO WS-DL-STATUS.

00305

IF IT01-PAY-TEST EQUAL SPACES

00306

MOVE IT01-PAY-TEST TO WS-DL-PAY-ERR.

00307

IF IT01-FICA-TEST EQUAL SPACES

00308

MOVE IT01-FICA-TEST TO WS-DL-FICA-ERR.

00309

IF IT01-IRS-TEST EQUAL SPACES

00310

MOVE IT01-IRS-TEST TO WS-DL-IRS-ERR.

00311

PERFORM 0080-WRITE THRU 0080-EXIT.

00312

0070-EXIT.

00313

EXIT.


 POSSIBLE CAUSE: WS-LINE-COUNT IS NOT
INITIALIZED. CHECK DATA DIVISION.

00314

0080-WRITE.

00315

IF WS-LINE-COUNT GREATER THAN 19

00316

WRITE CRO1-PRINT-RCD FROM

00317

WS-READING-LINE-1

00318

AFTER ADVANCING CHAN1

00319

WRITE CRO1-PRINT-RCD FROM

00320

WS-READING-LINE-2

00321

AFTER ADVANCING 2 LINES

00322

MOVE ZERC TO WS-LINE-COUNT.

00323

MOVE QM01-SSN1 TO WS-DL-SSN1.

00324

MOVE QM01-SSN2 TO WS-DL-SSN2.

00325

MOVE QM01-SSN3 TO WS-DL-SSN3.

00326

MOVE QM01-NAME TO WS-DL-NAME.

00327

MOVE QM01-IN TO WS-DL-IN.

00328

MOVE QM01-ST-ADD TO WS-DL-ST-ADD.

00329

MOVE QM01-CITY TO WS-DL-CITY.

00330

MOVE QM01-JC-CODE TO WS-DL-JC-CODE.

00331

IF QM01-PAY NUMERIC

00332

MOVE QM01-PAY TO WS-DL-PAY.

00333

IF QM01-FICA NUMERIC

00334

MOVE QM01-FICA TO WS-DL-FICA.

00335

IF QM01-IRS NUMERIC

00336

MOVE QM01-IRS TO WS-DL-IRS.

00337

WRITE CRO1-PRINT-RCD FROM

00338

WS-DETAIL-LINE

00339

AFTER ADVANCING 2 LINES.

00087

05 FILLER

PIC X(1).

00088

FD OM-NEM-FILE
 BLOCK CONTAINS 5 RECCDS,
 RECORD CONTAINS 80 CHARACTERS,
 LABEL RECORDS ARE STANDARD,
 DATA RECORD IS OM01-NEM-RCD.

00089

00090

00091

00092

00093

01 OM 1-NEM-RCD.

00094

05 OM01-SSN.

00095

10 OM01-SSN1

PIC 9(3).

00096

10 OM01-SSN2

PIC 9(2).

00097

10 OM01-SSN3

PIC 9(4).

00098

05 OM01-NAME

PIC X(14).

00099

05 OM01-IN

PIC X(2).

00100

05 OM01-ST-ADD

PIC X(29).

00101

05 OM01-CITY

PIC X(14).

00102

05 OM01-JC-CODE

PIC X(1).

00103

05 OM01-PAY

PIC S9(5)V9(2).

00104

05 OM01-FICA

PIC S9(3)V9(2).

00105

05 OM01-IRS

PIC S9(5)V9(2).

00106

05 FILLER

PIC X(1).

WORKING-STORAGE SECTION.

77 WS-TRANS-READ-SW

PIC X(3)

VALUE 'ONN'.

00110

77 WS-CM-READ-SW

PIC X(3)

VALUE 'ONN'.

00111

00112

77 WS-READ-COM-SW

PIC X(3)

VALUE 'OFF'.

00113

00114

77 WS-LINE-COUNT

PIC 9(2).

00115

77 WS-NINES

PIC 9(9)

VALUE SSSSSSSSSS.

00116

00117

01 WS-PEADING-LINE-1.

00118

05 FILLER

PIC X(54)

00119

VALUE SPACE.

00120

05 WS-PRI-TITLE

PIC X(24)

00121

VALUE 'DAC MASTER UPDATE REPORT'.

00122

05 FILLER

PIC X(55)

00123

VALUE SPACE.

00124

01 WS-READING-LINE-1.

00125

05 FILLER

PIC X(5)

00126

VALUE SPACE.

00127

05 FILLER

PIC X(12)

00128

VALUE 'STATES'.

00129

05 FILLER

PIC X(18)

00130

VALUE 'SOC-SEC-NUM'.

00131

05 FILLER

PIC X(13)

US ARMY INSTITUTE OF ADMINISTRATION

Fort Benjamin Harrison, Indiana 46216

OPERATING SYSTEMS



Prepared for: ADPOC, NCOAC, P/AC

Prepared by: Software Division, Computer Science Department

SEPTEMBER 1978

STUDENT NAME: _____

CLASS NO. _____

FOR INSTRUCTIONAL PURPOSES ONLY

TABLE OF CONTENTS

	PAGE
Introduction	1-1
Disk Operating System (DOS)	1-4.5
Initial Program Load (IPL)	1-5
Multiprogramming	1-5.4
Job Control Language	3-1
Compile/linkedit/execute	3-4
JOB card	3-5
OPTION card	3-5
USAIA standard options	3-10
EXEC card	3-11
Linkage Editor	4-1
PFASE card	4-1
INCLUDE card	4-2
ENTRY card	4-2
ACTION card	4-3
Linkedit map	4-3
Assignment of I/O devices	5-1
ASSGN card	5-4
ADAM	5-4.2
Data file specification	6-1
Tape labels	6-1
TLBL card	6-3
LBLTYP card	6-5
Disk files	7-1
DLBL card	7-2
EXTENT card	7-2
COBOL PE2/PE3 JCL exercise	7-7
Indexed Sequential (ISAM) files	7-8
Librarian	8-1
Service (list/punch) functions	8-3
Catalog functions	8-13
Rename Functions	8-17
Delete Functions	8-18
Condense functions	8-20
Update functions	8-21
Reallocate functions	8-25
Copy functions	8-26
DOS utilities	9-1
File-To-File Utilities	9-5
Utility control cards	9-7

TABLE OF CONTENTS (CONT)

Operating Systems (OS)	10-1
APPENDICIES:	
DOS Job Control flow chart	A
Compile/Linkedit/Execute diagram	B

HISTORY OF COMPUTERS:

The first type of computing machines that were used were the punch card machines. These machines used plugboards to store their programs. They were comparatively slow, had very limited capabilities and had no internal storage.

The PCM were followed by the first generation computers. These were big, relatively slow vacuum tube machines. All programming for these machines was done at the machine language level. They were single job initiation machines and were used mainly as scientific calculators.

About 1955-56, the second generation machines emerged. These had transistors instead of tubes and were faster than their first generation predecessors. It was during the second generation that compiler level languages such as COBOL and FORTRAN were developed. The second generation machines had a batch job capability and were the first of the general purpose machines. They were now used for Business Data Processing as well as scientific calculators.

Time marches on....about 1961, enter the third generation machines replacing the transistor with semiconductor circuits once again increasing the speed. Magnetic core was also being produced relatively cheaply so these computers had large core memories.

All of these changes did not come without some problems, most of which started in the second generation. The hardware technology was ahead of the software support so there was an inefficient use of the hardware. Large computers were not effective. Many of their resources such as memory and I/O devices sat idle a lot of the time. There was too much human intervention; too much interaction between man and machine. Even though there was too much interaction between man and machine, there was no way for man to interact with man thru the machine, in other words, there was no way to link one program to another.

The solution to these problems was an operating system. An operating system is an integrated set of programs designed to improve overall operating effectiveness. The purpose of an operating system is to increase the amount of data processed in a given amount of time. An operating system will not reduce the execution time for a single program. Over the years there have been many operating systems. One of the first was the Share Operating System (SOS) written by a group of IBM users. The SOS did not pan out. IBM produced IBSYS for its 7090 line of computers. This was about 1959. In 1961, Burroughs introduced the Master Control Program (MCP). It was the first of the third generation operating systems. It was not, however, until about 1963 when IBM announced DOS and OS that operating systems were really here to stay. There are three basic sources for operating systems. The first and least common is the user group, the time and expense (about \$60,000,000 for DOS) makes it almost impossible for a group of users to produce their own system. The second source is the Academic World. Universities often write their own special application systems using the wealth of research personnel on their faculties and any free graduate students. The last and most common source of operating system is the hardware manufacturer. Almost all operating systems come from this last source. There are three basic requirements for an operating system. The first is a control program to control all the systems' resources. The second is a language for man-system communication and finally a software library and some means for retrieving programs from that library. Not all operating systems will have all of these requirements in a very obvious form but these three needs will in some manner be present in every operating system.

Reduction in Costs: One of the first computers designed and built costs \$500,000, had 18,000 vacuum tubes, weighted 30 tons and was named "ENIAC". Today, a microprocessor can have the same computing power of ENIAC, can be the size of a baby's fingernail and cost \$10.

TYPES OF OPERATING SYSTEMS:

BPS (Basic Programming Support) - a minimum system resident in cards or tape for use in installations having no disk files. Support Assembler Language and RPG.

BOS (Basic Operating System) - A disk resident system supporting Assembler Language and RPG, with sufficient capacity to satisfy requirements of small installations.

TOS (Tape Operating System) - A tape resident system offering the capabilities of supporting all languages plus multiprogramming, storage protection, and the internal timer with no disk.

DOS (Disk Operating System) - a disk resident system for users requiring more advanced operational support than that of TOS. Supports all languages as well as remote processing, storage protection, multiprogramming, and the interval timer.

OS (Operating system) - a disk resident system that provides for the assignment of devices and auxiliary storage space at execution time. Provides for multiprogramming fifteen separate jobs at one time. Also provides an internal SPOOLing capability as well as a priority scheduling system.

OS/VS (Operating System / Virtual Storage) - a disk resident system that uses a Direct access storage device to simulate a larger memory capacity than the machine really has. For example a System 370/165 with 512,000 bytes of real memory can be made to perform as though it has 2,000,000 bytes of real memory.

OPERATING SYSTEM.

An Operating System - integrated set of programs designed to improve the total operating efficiency of a computer's operation.

Functions of the Operating System.

Scheduling of I/O operations

Protection of partition boundaries (required for multiprogramming)

Error handling as referenced to the five interrupts.

Machine check interrupt

Supervisor call (SVC) interrupt

Program check interrupt

External interrupt

I/O interrupt

Operator communication with CPU.

Provide for program loading from private or system core image library.

Provide for program termination.

REQUIREMENTS OF THE OPERATING SYSTEM.

A control program (supervisor)

Library facilities

Core image library

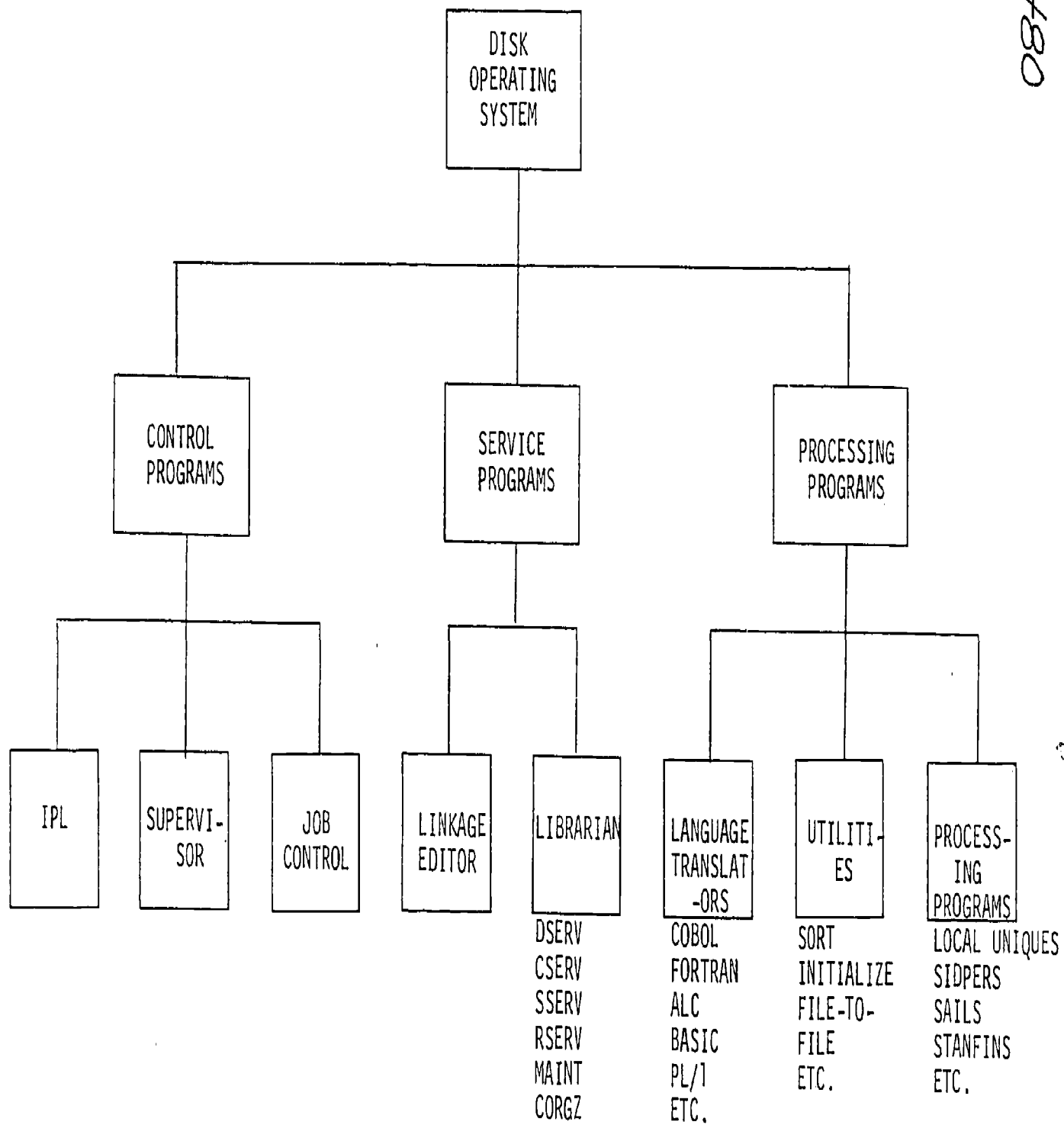
Source statement library

Relocatable library.

A device which provides a means for communication between the operating system and the computer operator (console typewriter).

NOTE: These libraries will be present in the operating system in some form.

1480



1-4.5

CSD.SFW.OS

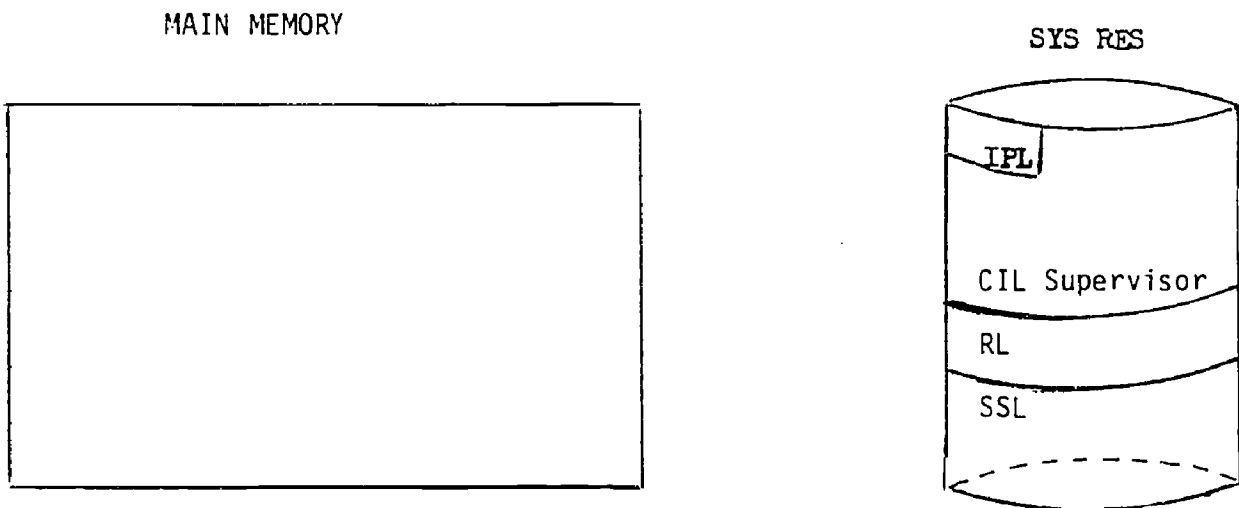
558

551

V. Types of programs which make-up the operating system.

a. Initial Program Load (IPL)

INITIAL PROGRAM LOAD



IPL - is loaded by the operator dialing the address of SYS RES on the CPU and pressing the console load key. The loading of IPL is a machine function, no instructions are executed. After the IPL is loaded, it is given control of the machine. At this time IPL clears main memory and the registers, loads the supervisor in the low core of main memory and passes control to the supervisor. When these tasks are accomplished, IPL has no further usage.

At the time when the supervisor is given control of the machine, the computer operator sets the date and time of day to be used by the supervisor and Problem Programs. He may also add or change the I/O device assignments in the Logical Unit Block (LUB) and Physical Unit Block (PUB) of the supervisor. Finally, he can change the partition sizes of the partitions allocated to the machine by entering an ALLOC statement.

EXAMPLE:

OII0A Give IPL Control commands

SET DATE = DD/MM/YY, clock=HH/MM/SS

OI20I DOS IPL Complete

At this point control is given to the control program (Supervisor)

The supervisor loads the Job Control Program into the Background partition. Job control finds the standard assignment for SYSRDR in the LUB and PUB. Next it reads the job control statements from the card reader (SYSRDR) performs the following functions:

1. Assigns device addresses to symbolic units.
2. Set up fields in the communication region.
3. Edit and stores label information.
4. Prepares for restarting checkpoints programs.
5. Clear the problem area to binary zeros between job steps.
6. Finally, it passes the name of the program to be executed to the System Loader.

7. Control is then passed to the Supervisor.

When the supervisor is given control from the Job Control Program, the System Loader goes to System Residence Pack (SYSRES), Core Image Library and fetches the program to be executed by placing it into main memory. Next the address of the first executable instruction is stored in the Control Unit of the CPU.

Finally, the problem program is given control and begins execution.

The preceding process explains what happens in the computer to start a problem program executing in the Background (BG) partition.

In order to start jobs in the Foreground 1 (F1) and Foreground 2 (F2) partitions, the operator must press the request key on the console typewriter and enter the following statements:

```
AR  BATCH F1
FI  ASSGN  SYSXXX,X'CUU'
PRESS THE REQUEST KEY
AR  BATCH F2
F2  ASSGN  SYSXXX,X'CUU'
```

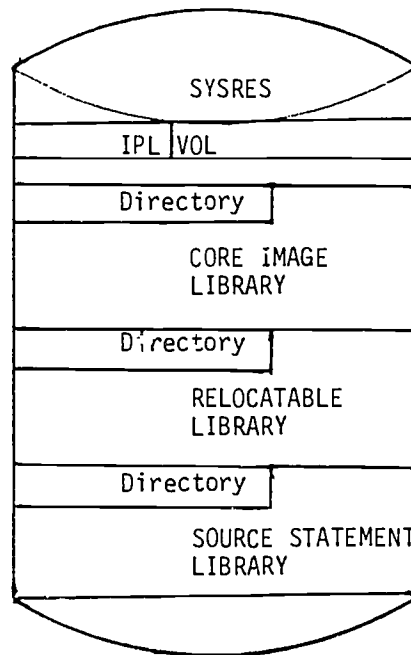
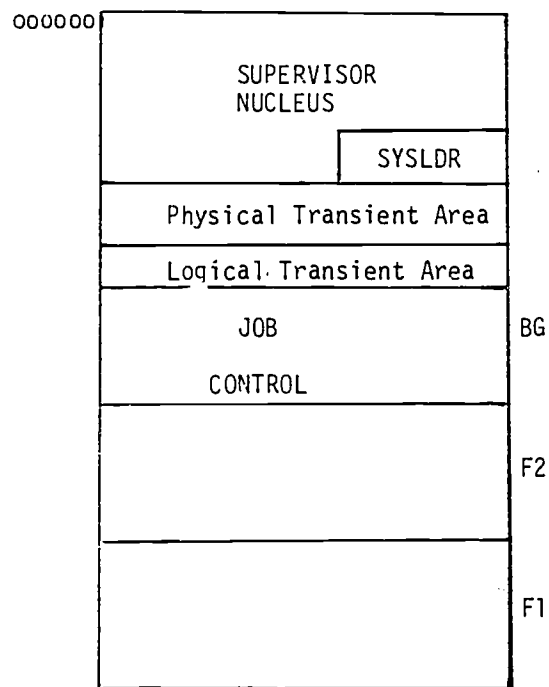
This process caused the Job Control Program to be loaded in the F1 and F2 partitions. Once Job Control is given control, it repeats the steps indicated above which cause the problem program to begin executing in each partition.

The Job Control Program is written over in the partitions by the problem program being loaded into the partition by the System Loader. When the problem program ends execution (Stop Run), control is given to the Supervisor. The supervisor reloads the Job Control Program into the partition of the program that ended execution wiping out the program. Job Control then reads SYSRDR to obtain the Job Control statements for the next task to be performed. This process continues until all jobs in the input have been processed through the computer.

484

JOB LOADING AND EXECUTING

MAIN MEMORY

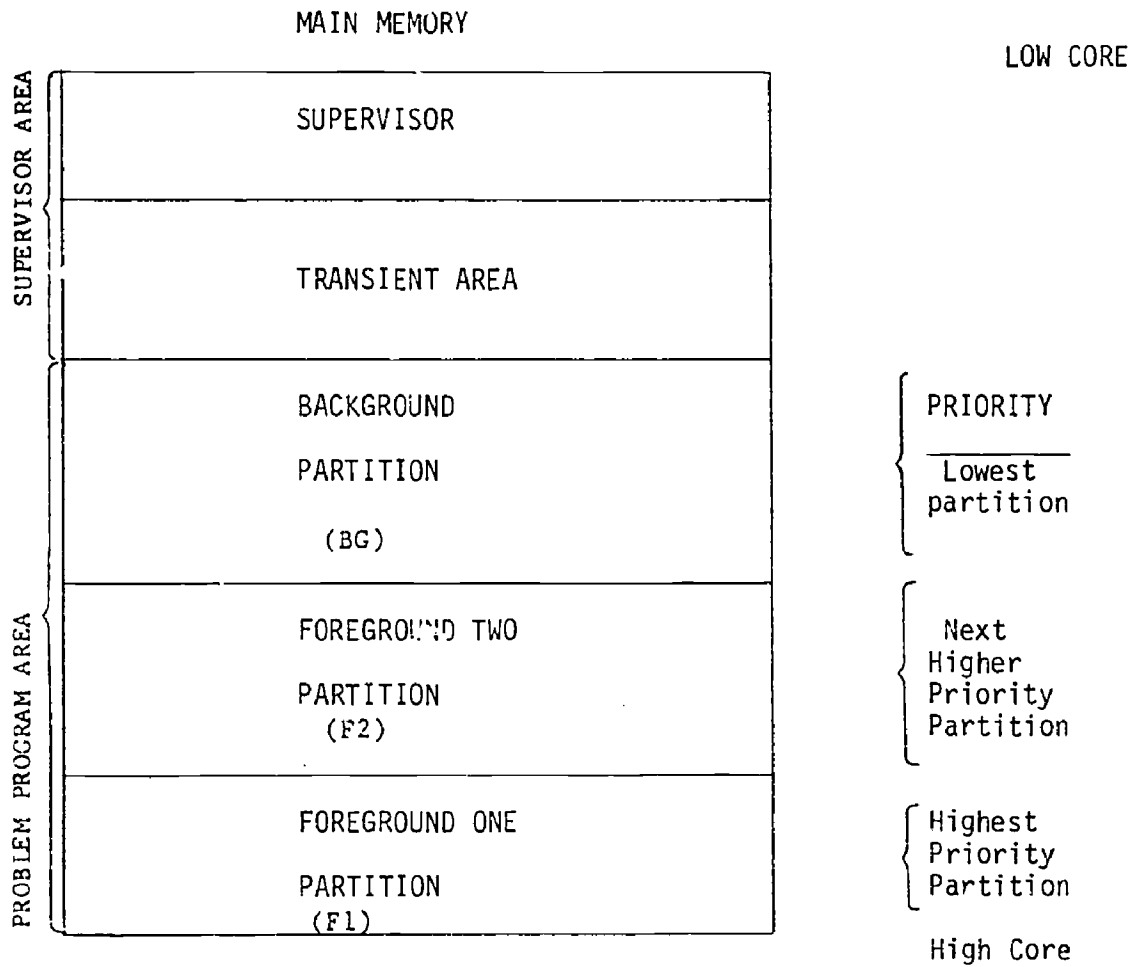


1-5.3

CSD.SFW.OS

502

MULTIPROGRAMMING - concurrent excution of two or more programs simultaneously residing in core storage of a computer.



b. Control Program is the backbone of the operating system and is composed of the following components:

(1) Job Management - identifies the job to be done and assigns I/O devices for the problem program.

(2) Task Management (supervisor) brings in the problem program from the library and responds to any error conditions by unloading one program and going on to the next one.

c. Data Management - handles movement of data between main storage and I/O devices. This also includes:

(1) Scheduling and programing the operation of channels.

(2) The identifications of data by volume (the device in which data is stored such as a reel of tape, or a disk pack)

(3) The resolution of error conditions that occur during the movement of data.

(4) Data Management is composed of the Input Output Control System, (referred to as IOCS)

(a) IOCS is a set of routines (programs) that handle the movement of data between main storage and I/O devices. These IOCS routines fall into two categories:

(1) Physical IOCS (PIOCS) - is composed of those I/O routines which supervise the reading and writing of data on I/O devices without regard for its logical content, format or organization.

(II) Logical IOCS (LIOCS) - Once the data has been brought from the I/O device by PIOCS, the LIOCS routines now operate on this data to make its logical content available to the problem program as required.

(b.) File Access Methods.

(I) Sequential access method - sequential processing is used to read/write and process successive records in a logical file.

(II) Index sequential access method - an index sequential file is built from sequential input and has indexes that permit individual records to be found for subsequent processing operations. By supplying the key of any record, a programmer can obtain the specific record randomly.

(III) Direct access method - records are usually organized in a random manner and are processed by referring to a record location reference supplied in the problem program.

This location reference indicates the exact record by track and key or identifier.

(IV) Basic Teleprocessing Access Method (BTAM) controls transmission and reception of messages over the communication lines in response to READ and WRITE macros. The LIOCS technique of accessing the file is not extended to the problem program.

(V) Queued Teleprocessing Access Method - has the same capabilities as BTAM with the addition of LIOCS technique of accessing the file by the problem program.

(c) Processing Programs.

(I) Language processor - are programs which use as input source programs and interpret them into a machine compatible language.

(a) FORTRAN compiler

(b) RPG compiler

(c) COBOL compiler

(d) PL/I compiler

(e) ALC assembler

(f) GPSS compiler

(d) Utility programs - generalized programs which, through the use of a control card, can perform a specific function.

- (I) Card to tape utility
- (II) Tape to print utility
- (III) Disk to print utility
- (VI) Sort/Merge ...etc.
- (V) Autotest.

(e) Service programs - generate the system, create and maintain the library sections, and edit programs into disk residence before execution.

(I) Link Editor Program - prepares programs for entry into the core image library by:

- (a) Assigning final main storage addresses to data and instructions.
- (b) Incorporating subroutines from the relocatable library, as requested by the programmer and combining separately written sections of the program into a single unified program.

(II) Librarian -Is a group of programs that maintains and reorganizes the disk library areas and provides printed and punched output from the libraries. Three libraries are used.

(a) Core Image Library. All programs in the system (IBM - supplied and user programs) are in machine code and loaded from this library by the system loader routine of the supervisor.

(b) Relocatable Library: This library stores object modules that can be used for subsequent linkage with other programs modules. A complete program of one or more modules can be placed in this library.

(c) Source Statement Library. This library stores IBM-supplied MACRO definitions and user defined source statement routines (such as MACRO definitions) built to provide extended program-assembly capability.

(III) Librarian Programs;

MAINT

CSERV

SSERV

RSERV

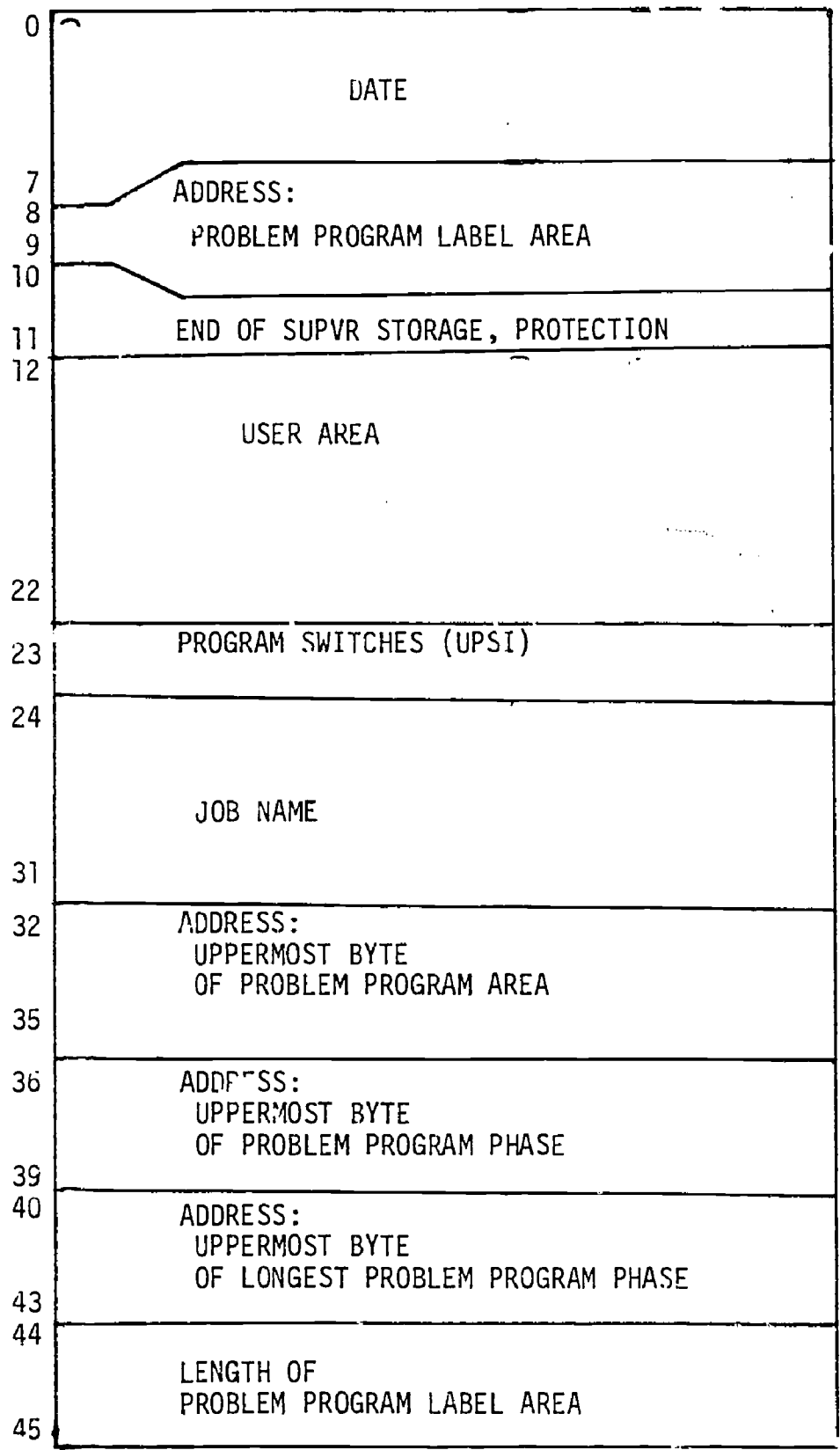
CORGZ

LISTVTOC

DSERV

DOS - MAIN STORAGE ORGANIZATION

SUPERVISOR	Minimum 6144 Bytes	PERMANENT ROUTINES	PERMANENT STORAGE - LOCATIONS PSW'S, CSW, CAW, ETC	
			I/O UNIT, CONTROL TABLES LUB, PUB, JIB	
			COMMUNICATIONS REGION - 46 BYTES	
			CHANNEL SCHEDULER	EXCP ROUTING I/O INTERRUPT RTE START I/O RTE
			OTHER INTERRUPT ROUTINES	SVC PROGRAM CHECK MACHINE CHECK EXTERNAL
			SYSTEM LOADER (SYSLDR)	
			RESIDENT (2311) ERROR RECOVERY ROUTINES	
			STORAGE PROTECTION (OPTIONAL)	
			TIMER SERVICES (OPTIONAL)	
			TRANSIENT ROUTINES	OPEN, CLOSE, DUMP, CHECKPOINT, OPERATOR COMMUNICATIONS, ERROR PROCESSING RTEs
PROBLEM: PROGRAM AREA (MINIMUM 10K BYTES)		USER PROGRAM or JOB CONTROL or LINKAGE EDITOR or LIBRARIAN ROUTINE or LANGUAGE TRANSLATOR		



The Communication Region

492

Diagram of PHYSICAL UNIT BLOCK (PUB) - Where the machine address of the I/O device can be found:

CHANNEL	BYTE 0
UNIT	BYTE 1
POINTER TO CHANNEL QUEUE	BYTE 2
ERROR RETRY COUNTER OR POINTER TO TAPE ERROR BLOCK (BOTH TEB AND TEBV)	BYTE 3
DEVICE TYPE	BYTE 4
DEVICE OPTIONS (TAPE SET MODE, ETC)	BYTE 5
CHANNEL SCHEDULER FLAGS	BYTE 6
JOB CONTROL FLAGS	BYTE 8

LOGICAL UNIT BLOCK (LUB) - Where system and programmer logical units are defined:

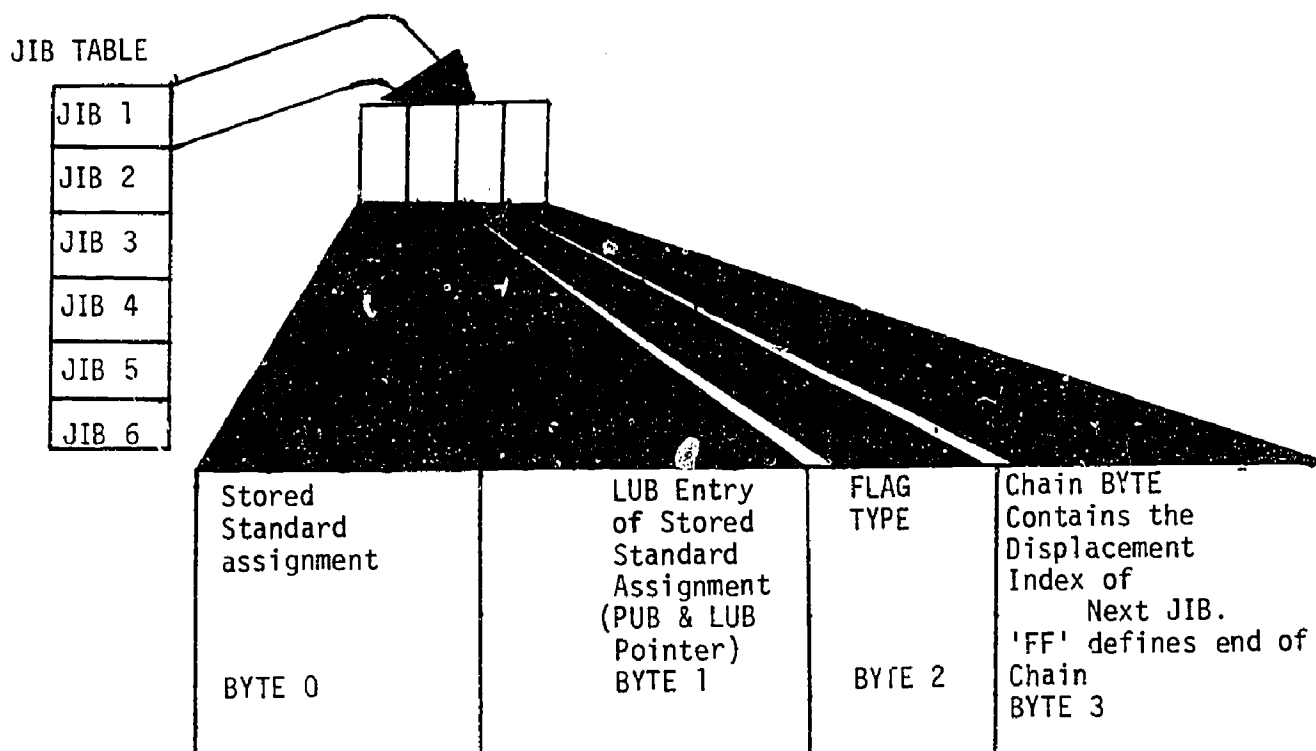
	BYTE 0	BYTE 1
Card Reader (JCL)	Pointer to Pub	Pointer to Job Information Block
Card Reader (DATA)		
Printer		
Console Type Writer		
Linkage Editor		
Work Area		
Disk Containing the Operating System		
Private Source Statement Library		
Private Relocatable Library		
Private Core Image Library		
Card Punch		
Programmer Logical Units		
	*SYSRDR	
	*SYSIPT	
	*SYSLSLST	
	*SYSLOG	
	SYSLNK	
	SYSRES	
	SYSSLB	
	SYSRLB	
	SYSCLB	
	*SYSPCH	
	SYS000 - SYSMAX	

* May be used in DTF of ALC

JOB INFORMATION BLOCK (JIB)

The JIB contains one of the following:

1. LUB entry of the standard assignment when a temporary LUB assignment is made.
2. PUB pointer for an alternate LUB assignment.
3. Extent information when DASD files protection is selected as a supervisor generation option.



494

JOB CONTROL LANGUAGE (JCL)

Whereas COBOL is a language for man-machine communication, JCL is a language for man-operating system communication.

JCL can enter the system through many channels. JCL cards may be keypunched by the programmer and placed into the system through the card reader along with such items as a COBOL source deck, a punched object deck, data cards or any combination of these. JCL may be cataloged in the Source Statement library to be invoked by SPOOLing programs. JCL may be entered into the system by the operator thru the computer console device.

STAGES OF PROGRAM DEVELOPMENT

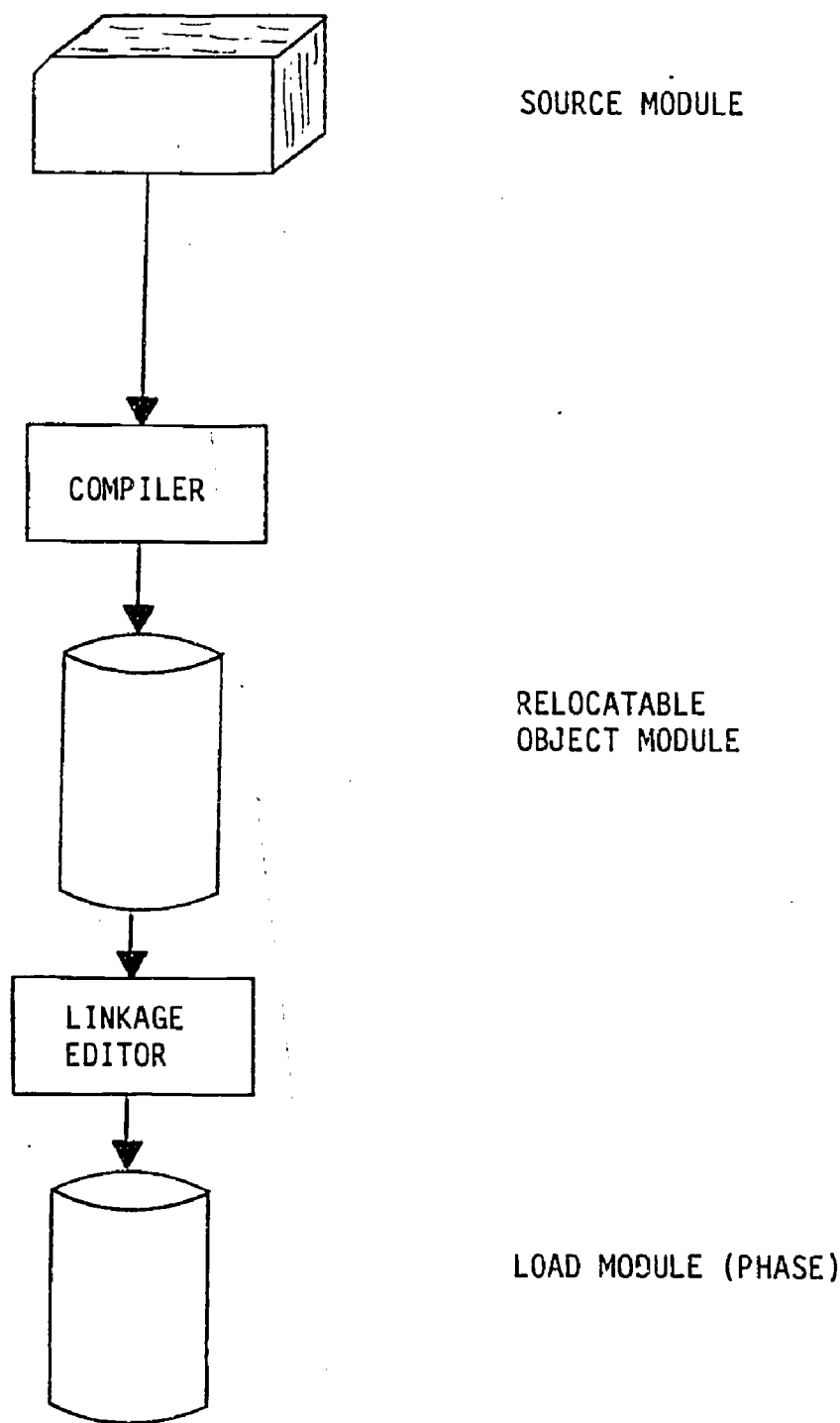
The term program is used to describe several things. The programmer codes sets of source statements that may be a complete program or part of a program. These source statements are then compiled into a relocatable machine-language program which, in turn, must be edited into an executable program, and may be combined with other programs. Consequently, it is convenient to refer to each stage of program development by a particular name.

A set of source statements that is processed by a language translator (Assembler, COBOL, FORTRAN, RPG or PL/I), is referred to as a source module or source deck (if in card form).

The output of a language translator is referred to as an object module (or deck). All object modules must be further processed by the linkage editor before they can be executed in the system.

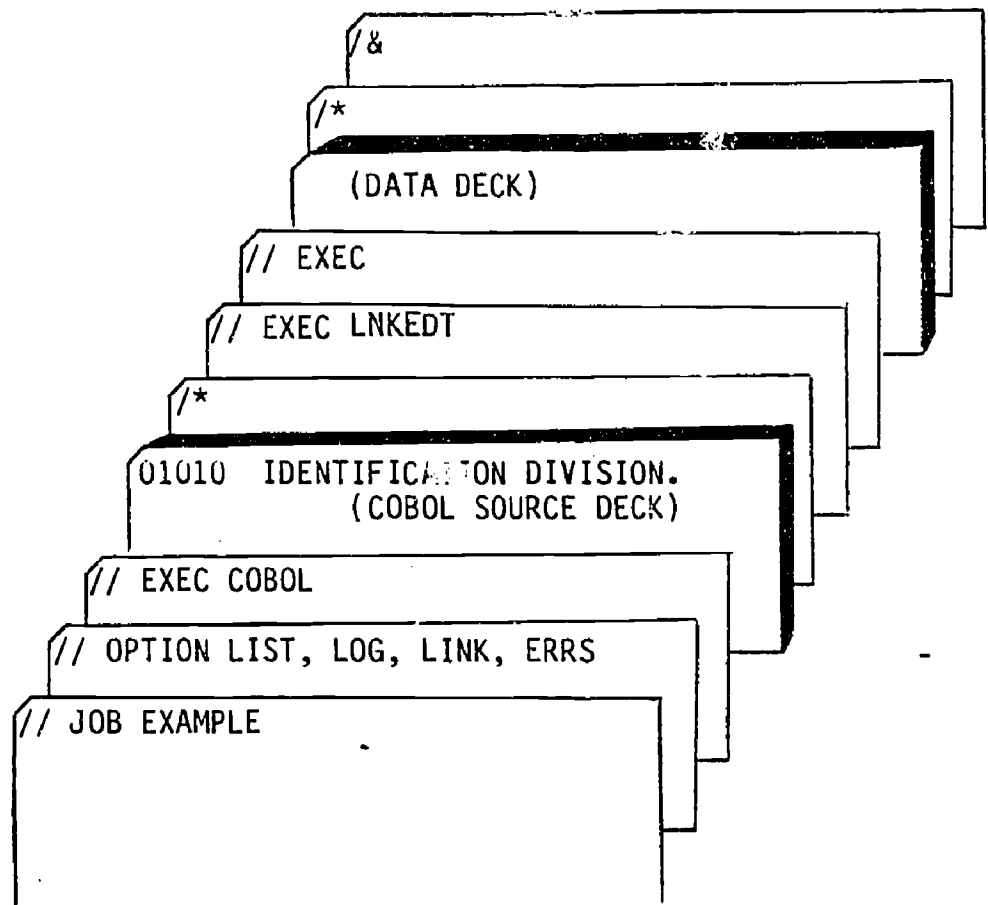
The output of the linkage editor consists of one or more program phases in the core-image library. A phase is in executable, nonrelocatable, core image form.

THREE PROGRAM FORMS



COMPILE - LINKEDIT - EXECUTE

The following represents the JCL for a very simple program that reads cards does some sort of calculations, and prints out a report. The cards bounded by the //JOB and the /& represent what is called a job. This particular job consists of 3 job steps each of which has a //EXEC card associated with it.



// JOB EXAMPLE

The JOB statement indicates the beginning of control information for a job.

FORMAT

// JOB jobname [accounting information]

jobname The name of the job. Must be 1 to 8 alphanumeric characters the first of which must be alphabetic.

accounting information
If the job accounting interface has been specified during system generation, 16 characters of user specified accounting information can be entered in the job statement. It must be separated from the jobname by a blank.

// OPTION LIST,LOG,LINK,ERRS

The OPTION statement specifies one or more of the Job Control options.

FORMAT

// OPTION option 1, option 2,.....

The purpose of this card is to override standard system options set at generation time. Selected options can be in any order. Options are reset to the system standards at the end of each job. (INDICATED BY & CARD)

LOG List all JCL on the printer from this card until a NOLOG, // JOB, or /& is encountered.

NOLOG Stop listing JCL.

DUMP Causes a dump of registers and main storage to be printed on printer if the program abnormally terminates.

NODUMP Turn off the dump option.

LINK Tells the COBOL compiler to put the object module in the linkage editor work area known as SYSLNK.

NOLINK Turn off link option.

DECK Tells the COBOL compiler to punch an object deck.

NODECK Turn off the deck option.

LIST Tells the COBOL compiler to write the source listing on PRINTER

NOLIST Turn off the list option.

```
00073      05 FILLER                                PIC X(82)
00074                                VALUE IS
00075      ' IS THE TOTAL NUMBER OF RECORDS PRINTED'.
00076      01 WS-PAGE-HEADING.
00077      05 FILLER                                PIC X(56)
00078                                VALUE IS SPACES.
00079      05 FILLER                                PIC X(7)
00080                                VALUE IS 'N A M E'.
00081      05 FILLER                                PIC X(8)
00082                                VALUE IS SPACES.
00083      05 FILLER                                PIC X(62)
00084                                VALUE IS 'SOC-SEC-NUM'.
00085      PROCEDURE DIVISION
00086      OPEN INPUT DISK-FILE
00087      OUTPUT PRINTER.
00088      PERFORM 0010-READ-AND-PRINT THRU 0010-EXIT
00089      UNTIL WS-DISK-EOF-SWITCH EQUAL TO '1'.
00090      MOVE WS-DISK-RCD-COUNT TO WS-TL-DISK-RCD-COUNT.
00091      WRITE OR01-PRINTER-RCD FROM WS-TOTAL-LINE
00092      AFTER ADVANCING 3 LINES.
00093      CLOSE DISK-FILE
00094      PRINTER.
00095      STOP RUN.
00096      0010-READ-AND-PRINT.
00097      READ DISK-FILE
00098      AT END
00099      MOVE '1' TO WS-DISK-EOF-SWITCH
```

500

LISTX Tells the COBOL compiler to write a PROCEDURE DIVISION MAP on SYSLST.

NOLISTX Turn off listx option.

86	000726		START	EQU	*	
	000726	41 10 C 045		LA	1,045(0,12)	LIT+13
	00072A	58 00 D 1C8		L	0,1C8(0,13)	DTF=1
	00072E	18 40		LR	4,0	
	000730	07 00		BCR	0,0	
	000732	05 F0		BALR	15,0	
	000734	50 00 F 008		ST	0,008(0,15)	
	000738	45 00 F 00C		BAL	0,00C(0,15)	
	00073C	00000000		DC	X'00000000'	
	000740	0A 02		SVC	2	
	000742	41 10 C 045		LA	1,045(0,12)	LIT+13
	000746	58 00 D 1CC		L	0,1CC(0,13)	DTF=2
	00074A	18 40		LR	4,0	
	00074C	07 00		BCR	0,0	
	00074E	05 F0		BALR	15,0	
	000750	50 00 F 008		ST	0,008(0,15)	
	000754	45 00 F 00C		BAL	0,00C(0,15)	
	000758	00000000		DC	X'00000000'	
	00075C	0A 02		SVC	2	
	00075E	50 20 D 1C0		ST	2,1C0(0,13)	BL=2
	000762	58 80 D 1C0		L	8,1C0(0,13)	BL=2
88	000766	58 00 D 1E8		L	0,1E8(0,13)	VN=01
	00076A	50 00 D 1E4		ST	0,1E4(0,13)	PSV=1
	00076E	58 00 C 010		L	0,010(0,12)	GN=01
	000772	50 00 D 1E8		ST	0,1E8(0,13)	VN=01
	000776		GN=01	EQU	*	
	000776	58 20 C 014		L	2,014(0,12)	GN=02
	00077A	D5 00 6 007 C 04D		CLC	007(1,6),04D(12)	DNM=1-398
	000780	07 82		BCR	8,2	
	000782	58 10 C 004		L	1,004(0,12)	PN=01
	000786	07 F1		BCR	15,1	
	000788		GN=02	EQU	*	
	000788	58 00 D 1E4		L	0,1E4(0,13)	PSV=1

SYM

Tells the COBOL compiler to write a DATA DIVISION MAP on the printer

NOSYM

Turn off SYM option.

INTRNL NAME	LVL	SOURCE NAME	BASE	DISPL	DEFINITION	USAGE
DNM=1-074	FD	DISK-FILE	DTF=01			DTFCD
DNM=1-107	01	IM01-DISK-RCD	BL=1	000	DS 0CL80	GROUP
DNM=1-133	02	FILLER	BL=1	000	DS 3C	DISP
DNM=1-149	02	IM01-NAME	BL=1	003	DS 18C	DISP
DNM=1-168	02	IM01-SOC-SEC-NUM	BL=1	015	DS 0CL9	GROUP
DNM=1-197	03	IM01-SSN-PART-1	BL=1	015	DS 3C	DISP
DNM=1-222	03	IM01-SSN-PART-2	BL=1	018	DS 2C	DISP
DNM=1-247	03	IM01-SSN-PART-3	BL=1	01A	DS 4C	DISP
DNM=1-272	02	FILLER	BL=1	01E	DS 50C	DISP
DNM=1-291	FD	PRINTER	DTF=02			DTFPR
DNM=1-322	01	OR01-PRINTER-RCD	BL=2	000	DS 133C	DISP
DNM=1-348	77	WS-DISK-RCD-COUNT	BL=3	000	DS 4C	DISP-NM
DNM=1-375	77	WS-LINE-COUNT	BL=3	004	DS 3P	COMP-3
DNM=1-398	77	WS-DISK-EOF-SWITCH	BL=3	007	DS 1C	DISP
DNM=1-426	01	WS-DETAIL-LINE	BL=3	008	DS 0CL132	GROUP
DNM=1-453	02	FILLER	BL=3	008	DS 51C	DISP
DNM=1-472	02	WS-NAME	BL=3	03B	DS 18C	DISP
DNM=1-489	02	FILLER	BL=3	04D	DS 2C	DISP
DNM=2-000	02	WS-SSN	BL=3	04F	DS 0CL11	GROUP

XREF Tells the COBOL compiler to write the symbolic cross reference list on the printer.

NOXREF Turn off XREF option

CROSS-REFERENCE DICTIONARY

DATA NAMES	DEFN	REFERENCE					
DISK-FILE	00025	00086	00086	00093	00097	00097	
IM01-NAME	00038	00109					
IM01-SSN-PART-1	00040	00106					
IM01-SSN-PART-2	00041	00107					
IM01-SSN-PART-3	00042	00108					
PRINTER	00027	00086	00086	00091	00103	00110	
OR01-PRINTER-RCD	00046	00091	00091	00103	00103	00103	00110 00110
WS-DISK-RSD-COUNT	00048	00090	00101	00101			
WS-LINE-COUNT	00049	00102	00105	00112	00112		
WS-DISK-EOF-SWITCH	00051	00088	00099				
WS-DETAIL-LINE	00053	00110	00110				
WS-NAME	00056	00109					
WS-SSN-PART-1	00060	00106					
WS-SSN-PART-2	00063	00107					
WS-SSN-PART-3	00066	00108					
WS-TOTAL-LINE	00069	00091	00091				
WS-TL-DISK-RCD-COUNT	00072	00090					
WS-PAGE-HEADING	00076	00103	00103				
PROCEDURE NAMES	DEFN	REFERENCE					
0010-READ-AND-PRINT	00096	00088					
0010-EXIT	00113	00088	00100				

ERRS Tell the COBOL compiler to summarize all errors in the source program on the printer.

NOERRS Turn off the ERRS option.

CARD	ERROR MESSAGE
1	ILA1087I-W ' ON ' SHOULD NOT BEGIN IN AREA A.
1	ILA1087I-W ' LOG ' SHOULD NOT BEGIN IN AREA A.
1	ILA1129I-C ID DIV. HEADER MISSING OR MISPLACED. ASSUMED PRESENT.
1	ILA1004I-E INVALID WORD ON . SKIPPING TO NEXT RECOGNIZABLE WORD.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
1	ILA1120I-W COMMA NOT FOLLOWED BY SPACE. ASSUMED.
51	ILA1132I-E INVALID SYSTEM-NAME. SKIPPING TO NEXT CLAUSE.
53	ILA1132I-E INVALID SYSTEM-NAME. SKIPPING TO NEXT CLAUSE.
60	ILA1056I-E FILE-NAME NOT DEFINED IN A SELECT. DESCRIPTION IGNORED.
73	ILA1056I-E FILE-NAME NOT DEFINED IN A SELECT. DESCRIPTION IGNORED.

CATAL Causes the program being link edited to be cataloged into the core-image library.

The Standard System Options of Student jobs are as follows: NOLOG, NODUMP, NOLINK, NODECK, LIST, NOLISTX, NOSYM, ERRS, XREF.

```
// EXEC FCOBOL
```

The EXEC statement indicates the end of control information for a job step and that execution of a program is to begin.

FORMAT

```
// EXEC  proname
```

Proname Is the name of the program cataloged in the core image library.

What occurs here is that the proname is passed to the supervisor which goes to the core-image library directory, finds the program, loads it into memory, and passes control to it to begin executing.

NOTE: FCOBOL is the name of a COBOL compiler.

```
source deck
```

This is the COBOL source deck that is to be compiled.

```
/ *
```

This delimiter card must be the last statement of each input data file. It causes the system to return an end-of-file indication to the program utilizing the data file.

```
// EXEC LNKEDT
```

This execute statement causes the linkage editor to be loaded into memory and control passed to it. The linkage editor looks for its input on SYSLNK. The output from the linkage editor will be placed in the temporary portion of the core-image library unless the OPTION CATAL is specified.

```
// EXEC
```

This execute statement with no name in the operand field causes the program just placed in the temporary portion of the core-image library to be loaded into memory and executed.

```
Data deck
```

These are the data cards that are to be processed by the program that was just compiled and link edited.

```
/*
```

End of data cards.

```
/*
```

End of job - all options reset to system standards.

EXERCISES

1. Write the JCL to compile a COBOL program and produce only an object deck, a source listing, an error listing, and a cross-reference listing. Assume all options are set in the NO_ _ _ mode for all exercises in the course.
2. Write the JCL for a single job to compile two COBOL programs. The first should produce only an object deck and the second should produce only a source listing, an error listing, a cross-reference listing and a data division map.
3. Write the JCL to compile and execute a COBOL program. Produce a source listing, an error listing, a cross-reference listing and if the program abnormally terminates, a core dump should be provided.

THE LINKAGE EDITOR

The linkage editor combines object modules supplied in the input job stream, and/or newly compiled object modules on SYSLNK, and/or object modules from a relocatable library. It edits these modules into executable programs.

The linkage editor puts these executable programs into one of two places. If the option CATAL has been specified the program is cataloged as a member of a core-image library. If not the program is placed in the temporary portion of the core-image library.

The compile-linkedit-execute example showed the linkage editor taking a just compiled program and linking it into the temporary portion of the core-image library.

If a program phase is to be cataloged into the core-image library a linkage editor control card is required. The card is a 'PHASE' card and the format is as follows.

```

┌ PHASE    name,origin
└

```

Card column 1 must be blank.

PHASE The operation code

name 1 to 8 alphanumeric characters used to give a name to the phase being cataloged. When the program phase is executed at a later date this is the name to be used in the execute card. (// EXEC name

,origin This operand allows the user to specify the load address to be used for this program phase. The possible options are;

1. symbol [(phase)][\pm relocation]
2. * [\pm relocation]
3. S [\pm relocation]
4. ROOT
5. + displacement
6. F + address

1. Used when building a program overlay structure. Says to link this phase at the same address as a previously linked phase with optional relocation shift.

2. Used when more than one phase is being linked together. Says put this one right behind the one just link edited. In a single phase operation this is the same as 'S'.
3. Says link this phase to load at the beginning of the partition we are working in.
4. Used in overlay and says that this particular phase is not to be overlayed but is to remain in memory while program is executing.
5. Specifies an absolute address.
6. Used to link edit a program to run in a foreground partition while the linkage editor is executing in background.

If object modules from the relocatable library or object decks external to the system are to be link edited, another control card comes into play.

```

┌ INCLUDE      [modulename]

```

modulename Symbol name of the module as used when cataloged in the relocatable library. If the operand is omitted the system assumes that the module is in the input job stream immediately behind the INCLUDE card.

The third linkage editor control card is the ENTRY card and is in the following format:

```

┌ ENTRY          entrypoint

```

entrypoint Symbolic name of the address where control is to be passed when the program is loaded into memory.

Note: The ENTRY card is normally not needed. Job control will write an ENTRY statement when EXEC LNKEDT is read to ensure that an ENTRY statement will be present to halt linkage editing.

EXERCISES

1. You are handed a deck of cards and told that is is an object module received from Fort Knox DPI. Write the JCL necessary to execute the program.
2. Write the JCL needed to catalog the object module mentioned in exercise 1 into the core-image library. Call the program FKX512.
3. Write the JCL to execute the program cataloged in exercise 2.

ASSIGNMENT OF I/O DEVICES

In order for a program to run in a computer system some means must be available to ~~LINK~~ the program to the physical devices such as card readers, printers, and disk drives.

The programmer is given the latitude to utilize any symbolic device address he wishes, ranging from ~~SYS000~~ through ~~SYSmax~~. For example in a COBOL program you may choose to call the printer '~~SYS005~~' and the card reader '~~SYS006~~' and the disk unit you are using '~~SYS010~~'. The preceding is accomplished through the use of the COBOL SELECT statement.

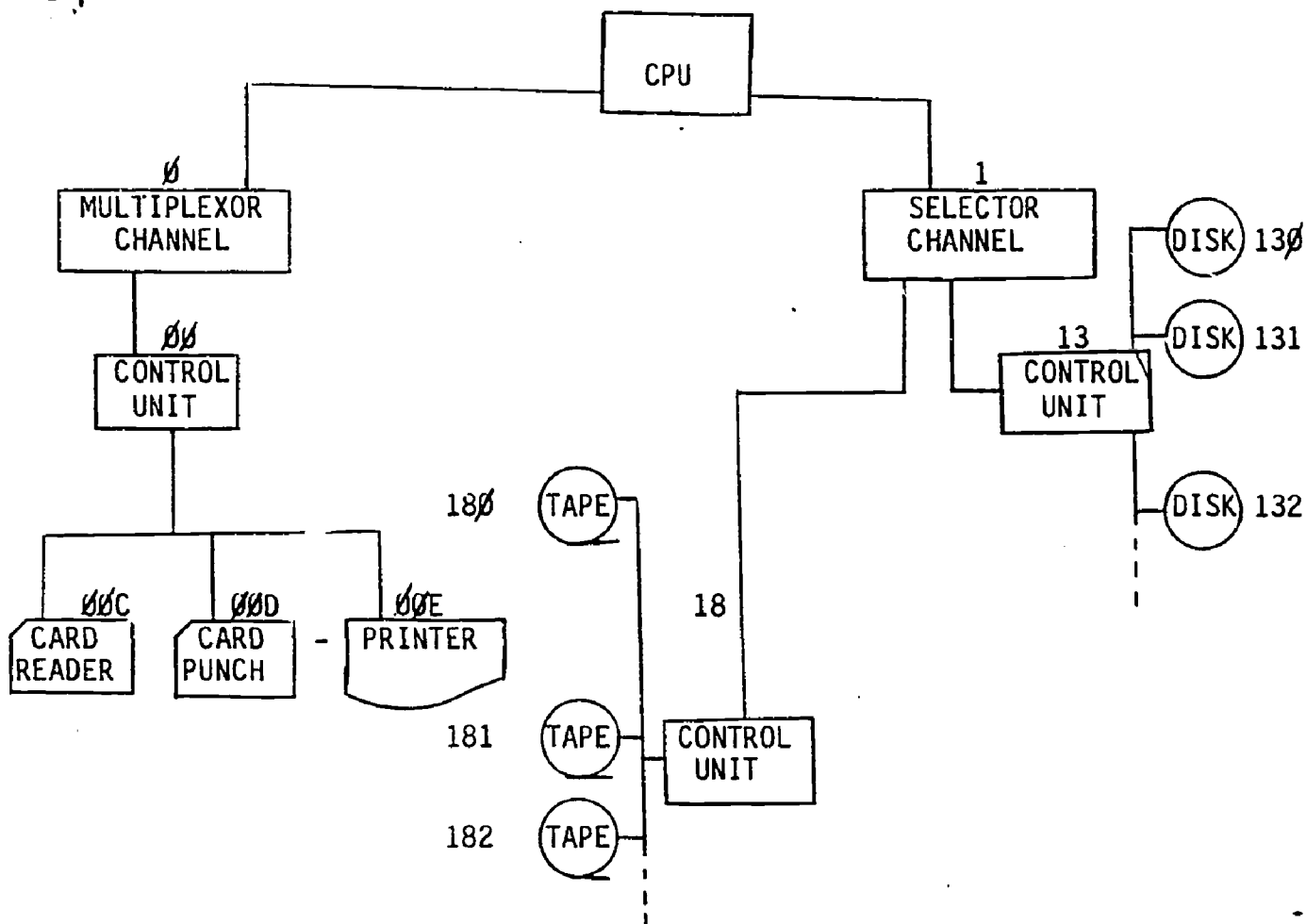
FILE-CONTROL

SELECT PRINT-FILE

ASSIGN TO ~~SYS005~~-UR-14~~03~~-S.

Because this program could be required to run on many different hardware configurations the physical device addresses are likely to be different. The following diagram depicts a typical hardware configuration indicating the physical device addresses.

NOTE: The // ASSGN card discussed later in this section serves the function of linking the program logical assignments to the hardware configuration.



There exists in the DOS system a set of fixed symbolic names used to reference various I/O devices. These are:

- SYSRDR - Card reader, magnetic tape unit, or disk extent (specific area) used for JOB control input.
- SYSIPT - Card reader, magnetic tape or disk extent used as input for programs.
- SYSPCH - Card punch, magnetic tape or disk extent used for punched output.
- SYSLST - Printer, magnetic tape or disk extent used for punched output.
- SYSLOG - Printer-keyboard or console printer-keyboard used for operator messages and to log JCL message.

SYSLNK - Disk extent used as input to the linkage editor.

SYSRES - System residence area on a disk drive.

SYSCLB - Disk extent used for a private core image library.

SYSRLB - Disk extent used for a private relocatable library.

SYSsLB - Disk extent used for a private source statement library.

SYSREC - Disk extent used to store error records collected by the system.

SYS~~000~~ - SYSmax - All other units on the system.

The first eleven of these symbolic names, called system logical units, are used by the system control programs and system service programs.

The remainder SYS~~000~~, SYS~~001~~,, SYSmax are known as programmer logical units. Some of these are defined and assigned to physical devices at system generation time such as the work areas required by the COBOL compiler.

The following are the physical device addresses for the ADMINcEN computer system and some of the Standard logical assignments set at system generation time:

PHYSICAL ADDRESS	DEVICE	LOGICAL ASSIGNMENT
00C	CARD READER	SYSRDR, SYSIPT, SYS 020
00D	CARD PUNCH	SYSPCH, SYS 021
00E	PRINTER	SYSLST, SYS 022
01F	CONSOLE TYPEWRITER	SYSLOG
18 0 -185	6 TAPE DRIVES	NONE
13 0	DISK DRIVE	SYSRES, SYSREC
131	DISK DRIVE	SYSLNK, SYS 000 thru SYS 009
132	DISK DRIVE	SYSsLB, SYSCLB
133-136	DISK DRIVE	NONE

Each of the disk assignments has associated with it information telling the system where exactly on the disk pack the specified file resides. For example Disk drive 131 has eleven different areas on the pack assigned to SYSLNK and SYS~~000~~ thru SYS~~009~~, they all represent different areas some of which are used by the COBOL compiler for work areas.

```
// ASSGN SYS005,X'00C'
```

The ASSGN statement assigns a logical I/O unit to a physical device. It remains in effect until the next change in assignment or until the end of job.

```
// ASSGN SYSxxx,address { ,x'ss' }
                        { ,ALT }
```

SYSxxx The programmer logical unit used in your program.

address The physical device address in the form of X'cuu' which indicates the Channel and Unit number (in hexadecimal).
 CHANNEL (c)
 0 for multiplexor channel, 1-6
 for selector channels 1 thru 6.
 UNIT (uu)
 00 - FE (0 to 254) in Hexidecimal

x'ss' used to tell the system the physical characteristics of a tape unit such as bytes per inch and parity used. Normally this information is provided at system generation time and is not needed unless changes have been made.

ALT Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. The specifications are the same for primary and alternate tape drives.

```
// ASSGN SYS010,X'180'
// ASSGN SYS010,X'181', ALT
```

// ASSGN cards must be placed in the job stream prior to the // EXEC that utilizes them and should be placed after the previous // EXEC in the job stream. The reason for the latter is to group all of the associated JCL with the Job step. In the case of multistep jobs JCL changes are difficult if all of the 'ASSGN' cards are bunched together at the beginning of the first step.

AUTOMATIC DEVICE ASSIGNMENT METHOD (ADAM)

The DOS ASSGN that we have just discussed does have some limitations. One of the limitations is that the JCL, which is written ahead of time, is hardware system dependent and would have to be modified card-by-card if the same set of JCL were to be used on a different hardware configuration. This does not present a great problem for programs that are written and run on the same machine. But, it does present a problem for Army Standard systems that are written in a central location along with the associated JCL and sent out to the world to be run on many different machines. A second limitation is the ability to get around an I/O device malfunction. It must be done in the same manner as above, a card-by-card change in the JCL ASSGN's.

To overcome these limitations ADAM was written to provide the ability to make DOS JCL hardware configuration independent and return system resources to the DOS console operator for management.

EXAMPLE: // ASSGN SYS010,X'TP0'

TP0 Positions 1 and 2 indicate the particular device type;

CR	card reader
CP	card punch
PR	printer
DK	disk drive
TP	tape drive
TD	tape drive w/dual density feature
T7	tape drive seven track

The 3rd position indicates the positional relationship of this device to any other device of the same classification. For example;
TP0,TP1,TP2,TP3,..... or CR0,CR1,PR0,PR1.....

There is a SYMBOLIC UNIT TABLE in memory that relates these symbolic assignments to the actual device addresses. The table logically appears as follows;

<u>SYMBOL</u>	<u>BG</u>	<u>F2</u>	<u>F1</u>
CR0	00C	00C	00C
CP0	00D	00D	00D
PR0	00E	00E	00E
TP0	180	182	184
TP1	181	183	185
TP2	182	184	180
TP3	183	185	181
TP4	184	180	182
TP5	185	181	183
DK1	130	134	136
.	.	.	.
.	.	.	.

In the example above we saw that the program called for 'TP0'. As you can see from the table if the program were run in the background (BG) partition it would be assigned to tape drive '180'. If it were run in foreground 2 (F2) it would be assigned to tape drive '182'. On some other hardware configuration it may be an address that does not even appear in this table.

516

If device 180 were to become inoperative for some reason, the operator would change the assignment in the symbolic unit table with the following three commands entered thru the console device;

```
SYMCHG TP0,BG=185  
SYMCHG TP4,F2=183  
SYMCHG TP2,F1=181
```

This is certainly easier than hunting up all of the // ASSGN SYSxxx,X'180' cards and repunching them.

The Symbolic Unit Table takes about 30 seconds to build at IPL time.

EXERCISES

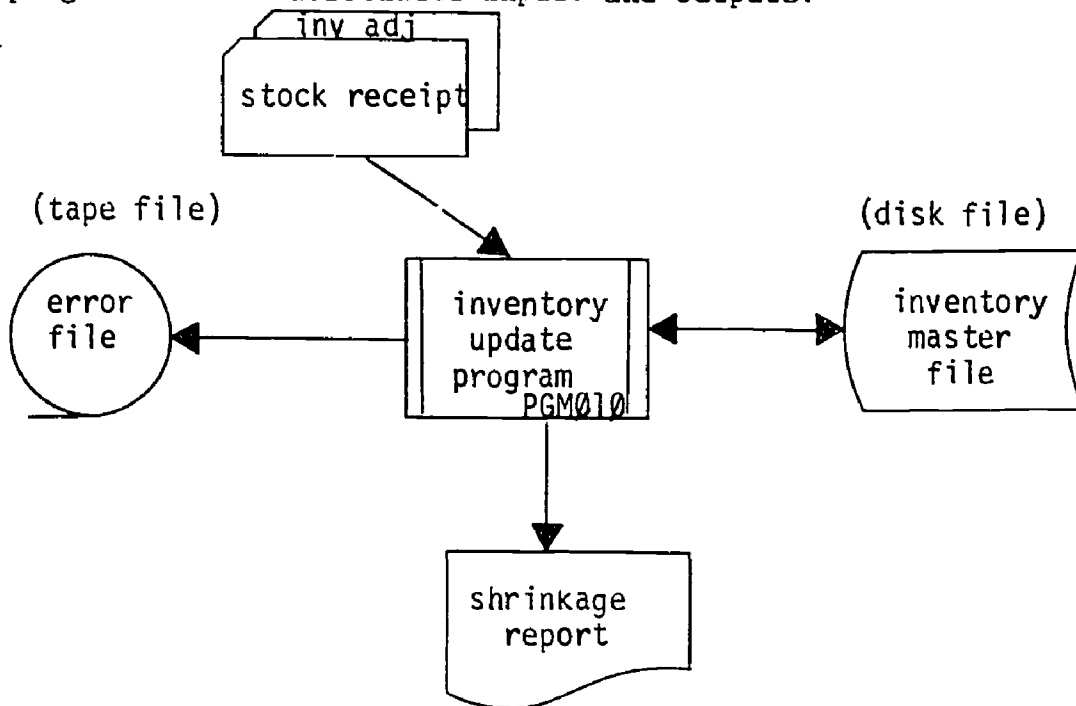
1. Write the JCL to compile and execute a COBOL program which uses the following SELECT statements

```
SELECT CARD-FILE ASSIGN SYS008-UR-2540R-S.  
SELECT PRINT-FILE ASSIGN SYS009-UR-1403-S.
```

2. Write the JCL to execute an object deck for a program which reads cards and produces a magnetic tape file and a printed report. The programmers logical units used by the program are SYS005 for the card reader, SYS008 for the printer, and SYS010 for the tape file.
3. Write the JCL for two COBOL programs which are to be compiled and executed as sep rate job steps within the same job. The first program while executing reads a card file and punches an output card file. The second program reads another set of data cards and produces a printed report. The first program uses SYS005 for the card reader and SYS007 for the card punch. The second program uses SYS006 for tne card reader and SYS005 for the printer.

DATA FILES

The diagram below is a portion of a Systems Flow Chart. It depicts a single program and its associated inputs and outputs.



We have already seen how the // ASSGN ... cards are used to tell the operating system what devices are to be assigned to this program in order to have all of these input and output files on line. No further discussion is necessary for the CARD and PRINT files. However, because there are normally hundreds or even thousands of reels of tape and hundreds of disk packs in an organization, it is necessary to have some way of uniquely identifying each disk pack and each tape reel. The problem is compounded by the fact that there may be more than one file on each volume.

TAPE FILES

(IBM TAPES LABELS FOR DOS; ORDER NO. GC24-5070)

Tape files may appear as one or more files per volume or more than one

volume per file. In any case the system must have some means of identifying the volume that it is reading and some means of identifying the file on that volume(s). This is accomplished through the use of STANDARD TAPE LABELS. Each volume has a standard volume label and a standard end-of-volume label. Each file has a standard header label and a standard end-of-file label. (See Page 6-2.1 Volume Layouts)

VOLUME-LABEL

The VOL1 label is written by a utility program, Initialize Tape. It is generally written once, when the reel of tape is first received in an installation. At that time, a permanent volume serial number is assigned to the reel and written on it as part of the volume label. This provides a permanent identification of the reel, as long as it is used for files, with standard labels. (See Page 6-2.2 Standard Volume Label)

FILE-HEADER-LABEL

Each tape file has a header label as the first physical record. It consists of 80 bytes of information as provided by the user and the IOCS routines to uniquely identify the file. (See Page 6-2.3 Standard File Header Label)

END-OF-FILE OR END-OF-VOLUME LABEL

Either one or the other of these will appear at the end of each volume depending on the file-volume relationship. The data in these labels are the same as that in the File-Header-Label with two exceptions. Bytes 1 thru 4 will contain either EOF1 or EOV1 and bytes 55 thru 60 contain a block count which indicates the number of physical records written on it's particular volume.

VOL1	HDR1	TM	DATA RECORDS	TM	EOF1	TM	TM
------	------	----	--------------	----	------	----	----

SINGLE-FILE-SINGLE-VOLUME

NOTE: TM is tape mark which is a special character that the tape drive writes on tapes to indicate check points.

VOL1	HDR1	TM	DATA RECORDS	TM	EOV1	TM
------	------	----	--------------	----	------	----

FIRST VOLUME

VOL1	HDR1	TM	DATA RECORDS	TM	EOV1	TM
------	------	----	--------------	----	------	----

MIDDLE VOLUME(S)

VOL1	HDR1	TM	DATA RECORDS	TM	EOF1	TM	TM
------	------	----	--------------	----	------	----	----

LAST VOLUME

SINGLE-FILE-MULTIVOLUME

VOL1	HDR1	TM	DATA RECORDS FILE A	TM	EOF1	TM	HDR1	TM	DATA RECORDS FILE B	TM	EOF1	TM	TM
------	------	----	---------------------	----	------	----	------	----	---------------------	----	------	----	----

MULTIFILE-SINGLE-VOLUME

1	4	5	1	1	8
			0	1	0

<u>BYTES</u>	<u>ENTRY</u>
1-4	VOL1
5-10	volume serial number written when tape is initialized by Utility Program.
11-80	not used for DOS

STANDARD FILE HEADER LABEL FOR DOS TAPE FILES.

VERSION NUMBER OF GENERATION									
HDR1	FILE IDENTIFIER	FILE SERIAL NUMBER	VOLUME SEQUEN NUMBER	FILE SEQUEN NUMBER	GENERA NUMBER	CREAT -ION	EXPIRA -TION	NOT USED	
1	45	22	22	33	33	34	44	55	8
		12	78	12	56	90	12	78	34

522

BYTES	ENTRY	
1-4	HDR1	
5-21	File Identifier	Permits the user to identify his logical file by an application-oriented unique name.
22-27	File Serial Number	Contains the volume serial number from the volume label of the first or only volume.
28-31	Volume Sequence Number	Used for multivolume files to insure that the tape reels are mounted in the proper sequence. For Single volume files this entry defaults to 0001.
32-35	File Sequence Number	Used for multifile volumes. For single volume files this entry defaults to 0001.
36-39	Generation Number	Identifies the various editions of a file, such as grandfather-father-son. relationship. Defaults to 0001 or number provided by user.
40-41	Version Number of Generation	Provides a more detailed identification of the editions of a file. Defaults to 01 or number provided by user.
42-47	Creation Date	System writes this date from IPL date entered by the operator.
48-53	Expiration Date	The date the file can be destroyed. Defaults to creation date or date as provided by user.
54-80	Not used for file header label.	

603

// TLBL TAPEFLE,'TAPE MASTER',150,771834

The tape label card is used to provide file label information for tape label checking and writing.

FORMAT

// TLBL filename,['file-id'],[date],[file-serial-number],
 [volume-sequence-number],[file-sequence-number],
 [generation-number],[version-number]

filename This operand serves to link the TLBL card to the COBOL program file description (FD). It consists of 1 to 7 alphameric characters the first of which must be an alphabetic.

EXAMPLE 1: SELECT TAPE-FILE1
 ASSIGN TO SYS011-UT-2400-S-TAPEFLE.
 // ASSGN SYS011,X'183'
 // TLBL TAPEFLE,'TAPE FILE ONE'

EXAMPLE 2: SELECT TAPE-FILE2
 ASSIGN TO SYS012-UT-2400-S.
 // ASSGN SYS012,X'184'
 // TLBL SYS012,'TAPE FILE TWO'

'file-ID' This optional entry consists of 1 to 17 alphameric characters, contained within apostrophes. It appears in the TLBL and the file label and serves to uniquely identify the file.

date This optional entry enables the user to specify how long the file is to be retained and can be specified in one of three ways:

1. Omit operand in which case the file will not be retained.
2. Enter an absolute expiration date in the form of yy/ddd.

84/005 file will expire on 5 January 1984

82/042 file will expire on 11 February 1982

3. Enter the number of days that the file is to be retained in the form of 0 to 9999.

9 file will expire 9 days after creation.

27 file will expire 27 days after creation.

9999 is used to specify a permanent file.

file serial

number This optional entry allows the user to specify the volume serial number of the first or only reel of tape in file.

volume

sequence

number If desired on input files you may wish to use this optional operand to access some volume of a multivolume other than the first. Enter 1 to 4 numeric characters. (This operand is usually omitted)

file

sequence

number This optional entry allows the user to access a specific file within a multifile volume. Enter 1 to 4 numeric characters.

Generation

number Optional operand for the user to specify a generation number.
 Enter 1 to 4 numeric characters.

Version

number Optional operand to sub-identify a particular generation.
 Enter 1 or 2 numeric characters.

// LBLTYP TAPE or // LBLTYP NSD(02)

The LBLTYP statement (reserve storage for label processing) defines the amount of main storage to be reserved at linkage-edit time.

FORMAT

// LBLTYP TAPE
 NSD(nn)

TAPE Used only if tape files requiring label information are to be processed, and no nonsequential DASD files are to be processed.

NSD(nn) Used if any nonsequential DASD files are to be processed regardless of other type files to be used. nn specifies the largest number of extents to be used in a single file.

The amount of storage that must be reserved for label information is:

1. For standard tape labels 80 bytes.
2. For sequential disk labels NONE.
3. For ISAM or DA disk file labels 84 bytes plus 20 bytes per extent.

EXERCISES

1. Write the JCL to execute a program that is cataloged in the core image library under the phase-name 'WS342' and contains the following select statements:

SELECT CARD-FILE

ASSIGN TO SYS008-UR-2540R-S.

SELECT TAPE-FILE

ASSIGN TO SYS011-UT-2400-S.

TAPE-FILE is an output file. Utilize 'ERROR FILE TAPE' as a unique file ID and put the file on tape SN 772206. Retain the file for one year.

2. Write the ASSGN's and TLBL's for a program containing the following Select statements:

SELECT TAPE-INPUT-FILE

ASSIGN TO SYS011-UT-2400-S.

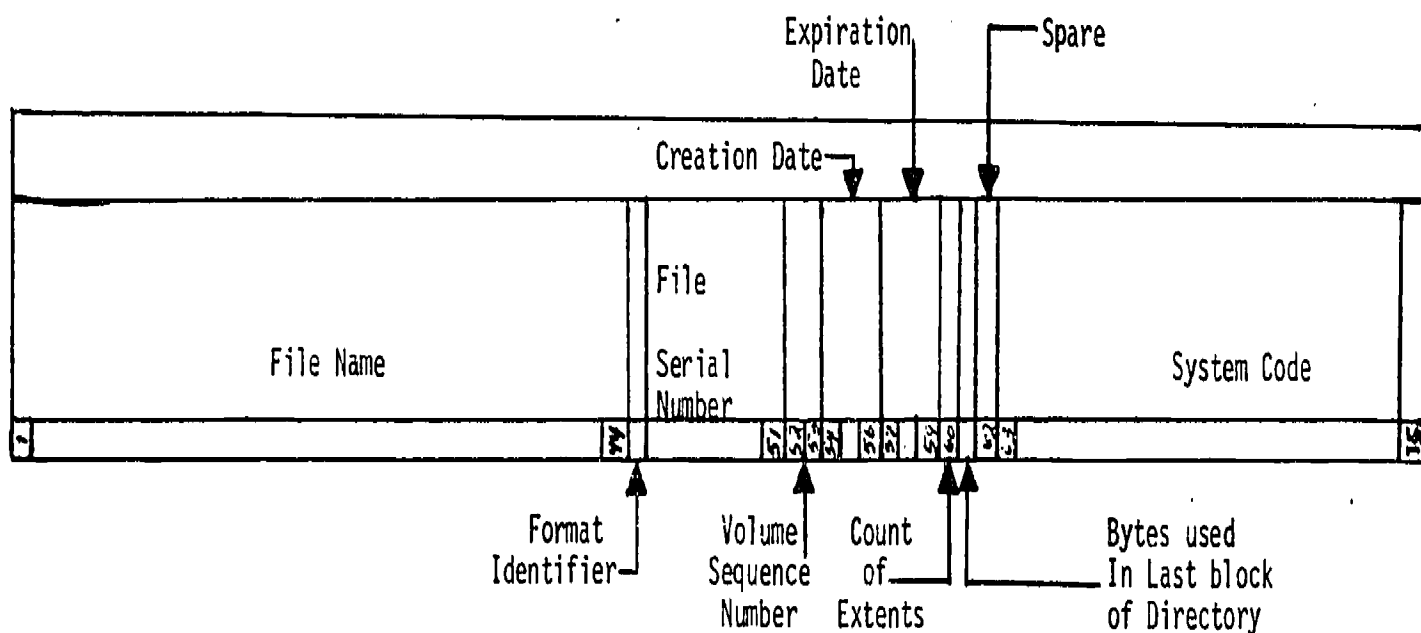
SELECT TAPE-OUTPUT-FILE

ASSIGN TO SYS012-UT-2400-S-NEWMST.

TAPE-INPUT-FILE has no unique file ID but is on tape SN 750349.

TAPE-OUTPUT-FILE is to go on tape SN 781433. Assign your own file ID. The file is to be retained indefinitely.

STANDARD MASS STORAGE DEVICE LABELS -- (VTOC)



Record Length

Option Code Key Location Space Remaining

										FIRST EXTENT			ADDITIONAL EXTENT			ADDITIONAL EXTENT			

```
// DLBL DISKFLE,'DISK MASTER FILE',87/130,SD
```

The disk label card is used to provide part of the label information for disk label writing and checking.

FORMAT

```
// DLBL filename,['file id'],[date],[code]
```

filename This operand serves to link to DLBL card to the COBOL program file description (FD). It consists of 1 to 7 alphanumeric characters, the first of which must be alpha.

```
EXAMPLE;          SELECT OM-DISK-MASTER-FILE
                   ASSIGN TO SYS007-UT-2314-S-DISK5.
```

```
// ASSGN SYS007,X'132'
// DLBL DISK5,'Mast.....'
```

file id This operand links the DLBL card to the data set on the disk pack. It is 1 to 44 characters in length the first of which must be alpha. The file id must be unique on any given disk pack.

date The date is used when creating an output file and can have one of two formats;
 yy/ddd using year and julian date to specify expiration date.
 dddd to specify the number of days the file is to be retained.

code This operand tells the system what kind of file this is;

<u>code</u>	<u>definition</u>
SD	Sequential disk file
DA	Direct access file
ISC	Indexed sequential (used only when creating file)
ISE	Index sequential (used for all other activities)

```
// EXTENT SYS007,123321,1,0,123,1520
```

The disk extent(area) card is used to tell the system such things as what disk pack the file is on, where on that pack does the file begin and how many tracks does the file occupy.

FORMAT

```
// EXTENT [symbolic unit],[serial number],[type],[sequence number],
           [relative track],[number of tracks]
```

symbolic unit This operand must match the symbolic unit in its associated assign card.

serial number Volume serial number of the disk pack for which this extent is effective.

type This operand specifies the type of extent this is;

<u>TYPE</u>	<u>PURPOSE</u>
1	Prime data area for any of the 3 file types.
2	Independent overflow for ISAM files.
4	Index area for ISAM.

sequence number For sequential disk files the first extent is sequence number 0, the second is 1, the third is 2 and so on.

For ISAM files the following rules apply;

0 Master index only
 1 Cylinder index only
 2 first prime data extent
 3 second prime data extent
 . etc..... 4,5,6....N
 N+1 Independent overflow

relative track This operand indicates the starting track of the file relative to the first track on the disk pack.

The user will normally know the cylinder and track number. If so, you will have to convert it to relative track using the following formula;

$$\text{relative track (RT)} = 20 \times (\text{cylinder number}) + (\text{track number})$$

EXAMPLE: Your sequential disk file begins on cylinder number 15 track number 5. What is the relative track?

$$\begin{aligned} \text{RT} &= 20 \times 15 + 5 \\ \text{RT} &= 305 \end{aligned}$$

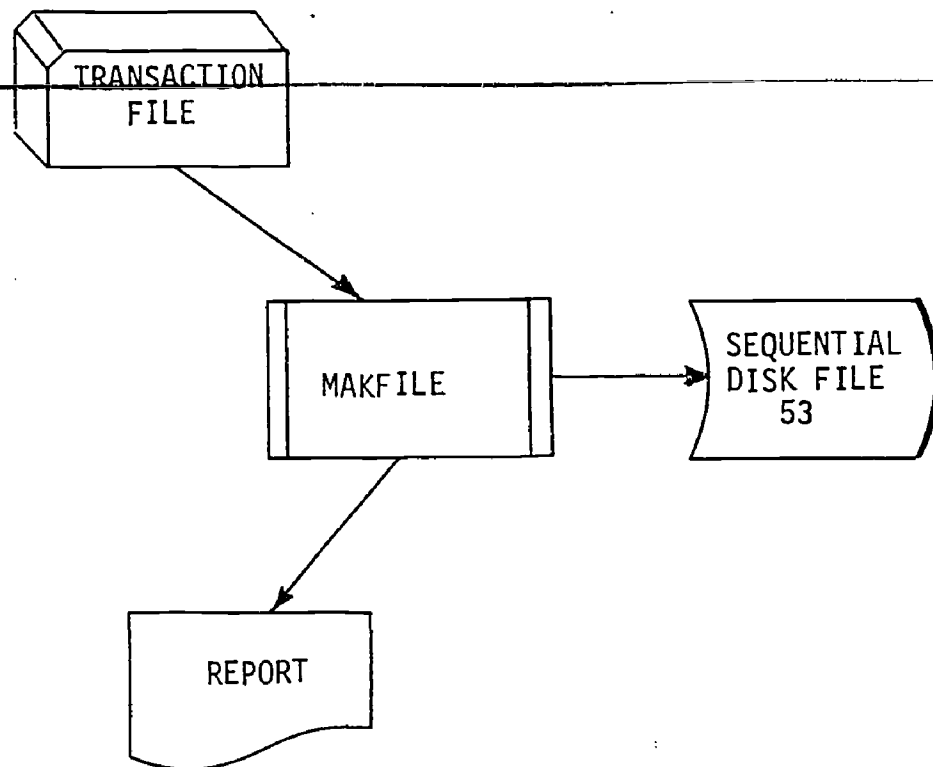
NOTE: For ISAM files prime data extents, the relative track must be a multiple of 20.

number of tracks This operand is the total number of tracks in this extent.

EXAMPLE: Your sequential disk file begins on cylinder 15 track 5 and ends on cylinder 21 track 15. What is the number of tracks?

$$\begin{aligned} \text{RT}(\text{start}) &= 20 \times 15 + 5 = 305, \quad \text{RT}(\text{end}) = 20 \times 21 + 15 = 435 \\ \text{number of tracks} &= \text{RT}(\text{end}) - \text{RT}(\text{start}) + 1 \quad 435 - 305 + 1 = 131 \end{aligned}$$

Requirement; You are to write the job control language to execute the program called MAKFILE, which is currently cataloged in the core image library.



The sequential disk file is to be called SEQUENTIAL FILE 53 and is to be placed on disk pack number 555663 which is to be mounted on disk drive 132. The file begins on cylinder 105 track 3 and ends on cylinder 112 track 7. The file is to be retained for 20 days. The COBOL program selects are as follows;

```

FILE-CONTROL.
  SELECT IT-CARD-FILE
    ASSIGN TO SYS004-UR-2540R-S.
  SELECT OR-MAKFILE-REPORT
    ASSIGN TO SYS005-UR-1403-S.
  SELECT OM-DISK-FILE
    ASSIGN TO SYS010-DA-2314-S-SEQFLE.
  
```

532
SOLUTION:

```
// JOB  EXSEQ
// OPTION  LOG,DUMP
// ASSGN  SYS004,X'00C'
// ASSGN  SYS005,X'00E'
// ASSGN  SYS010,X'132'
// DLBL   SEQFLE,'SEQUENTIAL FILE 53',20,SD
// EXTENT  SYS010,555663,1,0,2103,145
// EXEC  MAKFILE

      INPUT DATA
      DECK

/*

/8
```

$$RT(start) = 20 \times 105 + 3 = 2103$$

$$RT(end) = 20 \times 112 + 7 = 2247$$

$$\text{Number of tracks} = 2247 - 2103 + 1 = 145$$

EXERCISES

1. Write the JCL for a program called 'MAKSEQ' that reads cards and creates a sequential disk file. The file is to be stored on disk pack 122013 which will be mounted on drive 134. It will be stored from cylinder 10 track 0 through cylinder 119 track 19. The file should expire 150 days after it is created. File-ID is CUSTOMER MASTER-FILE.

```
SELECT CUST-MASTER-FILE
      ASSIGN TO SYS012-UT-2314-S.
SELECT CARD-INPUT-FILE
      ASSIGN TO SYS005-UR-2540R-S.
```

2. Write the JCL for a card to disk program. The COBOL program is to be compiled and executed. The file is to be stored on disk pack 506012 which will be mounted on drive 132. It will be stored from cylinder 24 track 0 through cylinder 49 track 19. The file-ID is PRICE-MASTER-FILE. The file is to be retained indefinitely.

```
SELECT CARD-MASTER-FILE
      ASSIGN TO SYS026-UR-2540R-S.
SELECT PRICE-MASTER-FILE
      ASSIGN TO SYS024-UT-2314-S-PRIMAST.
```

COBOL PE2/PE3 JCL EXERCISE

COMPILE JØ1
EXECUTE JØ1



I. REQUIREMENT. Write the DOS JCL necessary to execute your COBOL PE number 2 or 3. You are expected to get the same compiler outputs as before as well as a clean execution of your program.

II. COBOL program changes. There are some differences between DOS and OS COBOL. In order to get a clean compile you will have to do the following:

1. Change the disk file assign to read
ASSIGN TO SYS006-UT-2314-S.
2. Change the printer file assign to read
ASSIGN TO SYS005-UR-1403-S.
3. Change the printer FD label clause to read
LABEL RECORDS ARE OMITTED
4. Remove the printer FD BLOCK CONTAINS clause.

III. METHODS.

1. Replace the // JOB name card with the following card filled out the same way as your normal computer run card;

LAST NAME, FIRST ROOM = nnnn <i>(please punch)</i>		
<p>U.S. ARMY INSTITUTE OF ADMINISTRATION COMPUTER SCIENCE DEPARTMENT</p> <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: 80%;"> FOR USE ONLY WITH EXERCISE JØ1 </div> <h1 style="margin: 10px auto; width: 150px;">DOS</h1> <p>FOR INSTRUCTIONAL PURPOSES ONLY</p>		<p>GENERAL JOB TICKET</p>  <p>ACADEMIC COMPUTER FACILITY</p>

2. Be sure all of your JCL is listed on the printer.
3. Do not use the DECK or CATAL options.
4. The disk pack is to be mounted on disk drive 'DK3'.
5. The output is to be printed on printer 'PRØ'.
6. The serial number of the disk pack is DOSWK3.
7. The file ID is 'SOFTWARE DATA FILE 14 * RESIDENT'.

INDEXED SEQUENTIAL ACCESS METHOD (ISAM) FILES

An ISAM file is basically a sequential file with a series of indexes that allow the processing of single records located anywhere within the file. An ISAM file can also be processed sequentially if desired. There are several different types of areas associated with an ISAM file and these are explained below (See diagram on the next page as you read).

PRIME AREA is where the records are located and may occupy one cylinder or many disk packs. The prime area is defined by the programmer thru JCL..The records are located in this area in sequential order (logically) by record key in ascending order. A KEY is some unique field within the logical record. In a personnel file perhaps we would use Social Security Number.

INDEX AREA is the place on the disk pack where ISAM builds and maintains a series of indexes used to access or add records to the file when processing in a random mode.

TRACK INDEX occupies the first track of each cylinder of the prime area. The track index is defined by ISAM and as with all other indexes is loaded by ISAM. Its' function is to tell ISAM on which track of this cylinder the record is located or if being added, on which track it is to be added to.

CYLINDER INDEX is defined by the programmer thru JCL. Its' function is to tell ISAM on which cylinder the record should be located.

MASTER INDEX is defined by the programmer using JCL. Its' function is to tell ISAM which track of the cylinder index to look for this particular record. The master index is the only optional index in the ISAM file. It is normally defined if the cylinder index is greater than 4 tracks.

OVERFLOW AREA is used when records are added to an already established file.

CYLINDER OVERFLOW. A certain number of whole tracks are reserved in each cylinder for overflow records from the prime area. The programmer may specify the number of tracks to be reserved by means of the APPLY CYL-OVERFLOW clause. If zero is specified no cylinder overflow will be reserved. If the clause is omitted, ISAM defaults to 20% of each cylinder. This area is not defined by JCL.

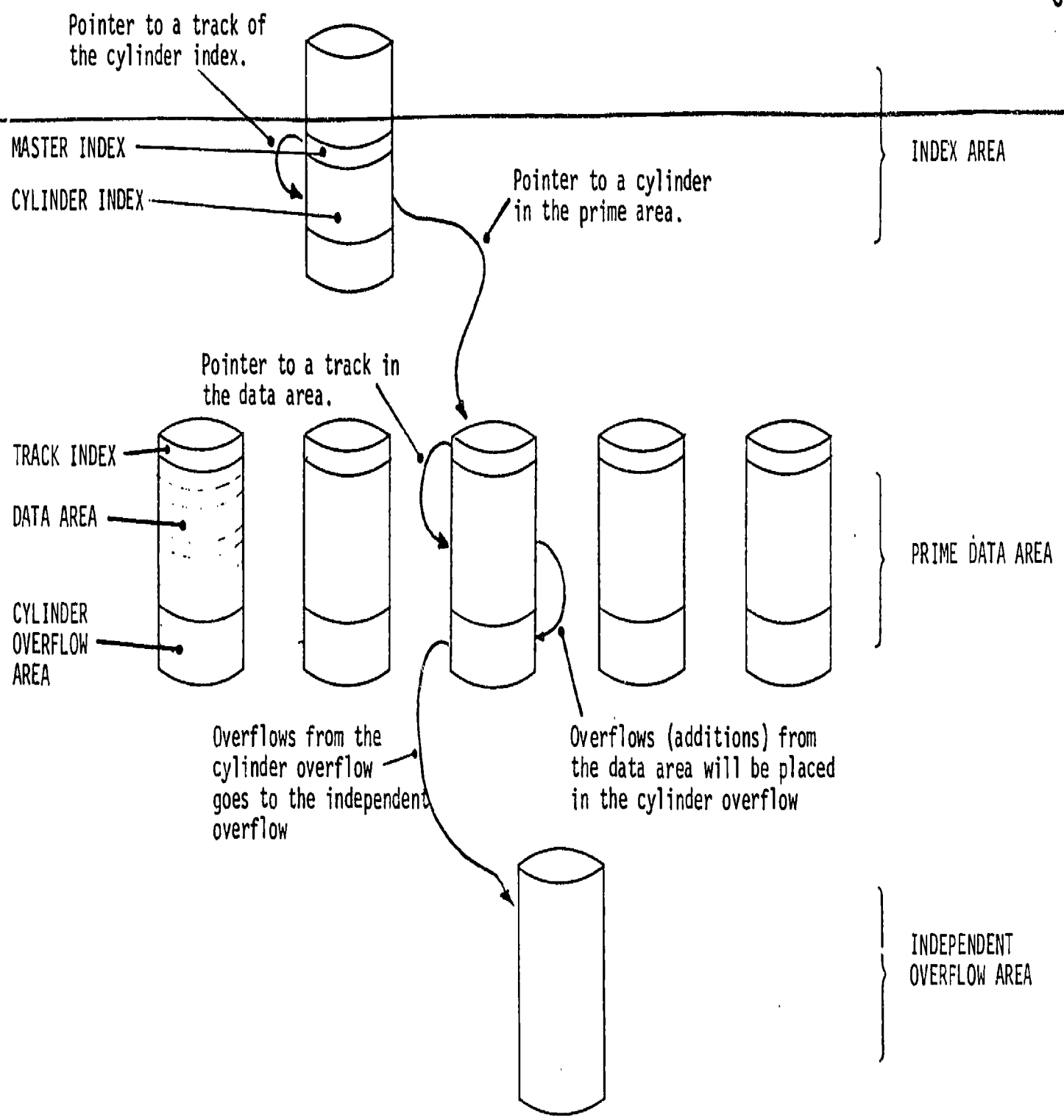
INDEPENDENT OVERFLOW AREA. Overflow records from anywhere in the prime area are placed in a certain number of cylinders reserved solely for this purpose. The size and location of this area is defined by the programmer thru JCL.

ELEMENTS OF AN INDEXED SEQUENTIAL (ISAM) FILE

536

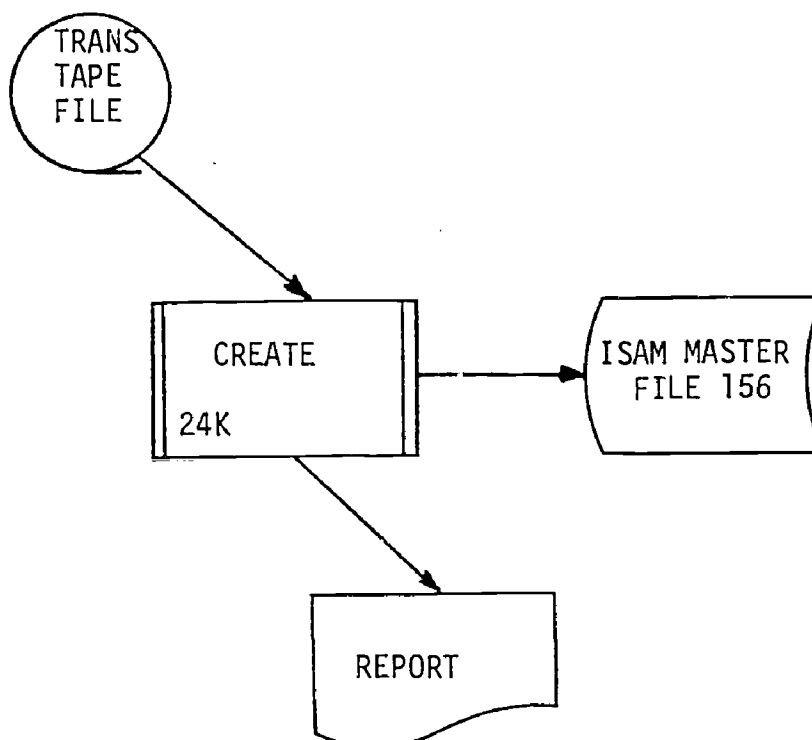
7-9

CSD.SFW.05



619

Requirement: You are to write the job control language needed to execute the program called CREATE, which is currently cataloged in the core image library. The systems flow chart below will assist you by showing you the relationships of all files.



The transaction tape file has a file Id of 'TRANS TAPE 156' and the file serial number is 556783. It is to be mounted on drive 182.

The ISAM file occupies 3 disk packs as follows;

The MASTER INDEX is on disk pack number 122375 and begins on cylinder 15 track 3 and ends on cylinder 15 track 5. Mount pack on drive 131.

The CYLINDER INDEX, as required, begins on the next track after the master index and takes 12 tracks.

The PRIME AREA is on disk pack number 155732 and occupies cylinders 5 thru 167. Mount that pack on drive 132.

The INDEPENDENT OVERFLOW AREA is on disk pack number 123723 and occupies all of cylinder 12. Mount that pack on drive 133.

Retain the file for 180 days and call it 'ISAM MASTER FILE 156'.

The COBOL program selects are as follows;

FILE-CONTROL.

```

SELECT IT-TRANS-TAPE-FILE
  ASSIGN TO SYS010-UT-2400-S-SQTAPE.
SELECT OM-MASTER-FILE
  ASSIGN TO SYS015-DA-2314-I-ISMAST
  ACCESS IS SEQUENTIAL
  RECORD KEY IS OM01-SSAN.
SELECT OR-MASTER-CREATE-RPT
  ASSIGN TO SYS005-UR-1403-S.
  
```

I-O-CONTROL.

```

APPLY CYL-OVERFLOW OF 2 TRACKS ON OM-MASTER-FILE
  
```

538

SOLUTION:

```
// JOB ISAMEX
// OPTION LOG,DUMP
// ASSGN SYS005,X'00E'
// ASSGN SYS010,X'182'
// TLBL SQTAPE,'TRANS TAPE 156',,556783
// ASSGN SYS015,X'131'
// ASSGN SYS020,X'132'
// ASSGN SYS025,X'133'
// DLBL ISMAST,'ISAM MASTER FILE 156',180,ISC
// EXTENT SYS015,122375,4,0,303,3
// EXTENT SYS015,122375,4,1,306,12
// EXTENT SYS020,155732,1,2,100,3260
// EXTENT SYS025,123723,2,3,240,20
// EXEC CREATE
/&
```

EXERCISES

1. Write the DLBL and EXTENT cards to create an ISAM file. The prime data area will be the area from cylinder 20 track 0 through cylinder 119 track 19. The master index will use cylinder 140 track 0 through cylinder 140 track 3. The cylinder index will use cylinder 140 track 4 through cylinder 140 track 15. The file is to be stored on pack 223556 and is to be saved for 500 days. Use the following select statement.

```
SELECT INVENTORY-MASTER-FILE  
      ASSIGN TO SYS009-DA-2314-I-MSTFLE.
```

2. Write the ASSGN, DLBL, AND EXTENT cards to create an ISAM file on disk pack 443556 drive 135. The file should not expire. The prime data area will be cylinder 29 track 0 through cylinder 35 track 19 and the cylinder index will be from cylinder 28 track 0 through cylinder 28 track 2. Use the following select statement.

```
SELECT CUSTOMER-MASTER  
      ASSIGN TO SYS005-DA-2314-I-CUSMSTR.
```

EXERCISES

3. Write the JCL to compile and execute a COBOL program which is to randomly retrieve and update the master file created in problem #2. The file-ID for the transaction file is TRANSACTION FILE and the serial number is 100005. No other information is needed for checking this label. Disk drive 134 and Tape drive 183 are to be used. Use the following COBOL select statements.

```
SELECT MASTER-FILE
      ASSIGN TO SYS021-DA-2314-I-CUSMSTR.
SELECT TRANS-FILE
      ASSIGN TO SYS022-UT-2400-S-TRNSFLE.
```


LIBRARIAN

As we have seen the DOS system consists of three libraries; the core image library, the relocatable library, and the source statement library.

We have also seen some of the functions performed on the libraries.

1. The linkage editor places the link edited phases into the core image library either in the temporary portion for immediate execution or catalog them as permanent members for execution at any time.
2. The system loader finds executable phases in the core image library and loads them into memory for execution.
3. Link editor will load modules into SYSLNK from the relocatable library to be link edited.
4. The linkage editor will extract relocatable IOCS modules from the relocatable library and attach them (link edit) to COBOL Programs.
5. The COBOL compiler and the assembler have the capability of extracting source code from the Source statement library and combining it with other source code.

FUNCTIONS OF LIBRARIAN

1. Maintenance functions add, delete, or rename components of the three libraries, condense or remove spaces that may exist in the directories or libraries themselves. Alter the amount of space allocated for each library. The program phase that performs this maintenance function is called MAINT and it applies to all three libraries.

2. Service functions will print and/or punch library members and directories. There are all four program phases that perform these services and they are:

DSERV - library directories (all)

CSERV - core image library

RSERV - relocatable library

SSERV - Source statement library

3. Copy functions will either completely or selectively; copy the disk pack on which the system resides, create private libraries, or merge libraries. The program phase that copies libraries is called CORGZ.

DISPLAY (SERVICE FUNCTION)

1. DSERV: the purpose of DSERV is to list, on the printer, the contents of the directories of the three DOS libraries in a readable format.

Format of control card:

% DSPLY directory [,directory]

directory: can be one of the following:

TD

CD

RD

SD

ALL

EXAMPLE: The following example would be used to cause the system to print out the directory for the core image library.

// JOB CORING

// EXEC DSERV

% DSPLY CD

/*

/&

PHASE NAME	DISK ADDR	NO. RCDS	SIZE LAST RCD	LOAD ADDR	ENTRY ADDR
---------------	--------------	-------------	------------------	--------------	---------------

-----DEC-----				-----HEX-----	
---------------	--	--	--	---------------	--

	C	H	R				
FLCRSEQE	034	18	03	001	0288	005000	005000
FLCRKONE	034	18	04	001	0704	005000	005000
U43ATP	034	19	01	007	0720	000000	000608
P23ATPLE	035	00	04	001	1601	000000	0005C0
P23SSPLE	035	01	01	007	0209	000648	002148
P23SPWLF	035	02	04	020	0205	002E90	00A6D8
PW4140K1	035	07	04	018	0409	000000	00A8D8
P23ATP	035	12	02	001	1378	0360C0	0360C0
SURSPQOL	035	12	03	006	1659	036628	038128
SURPOWER	035	14	01	015	0884	0380A0	03D7F8
P23ATP30	035	17	04	001	1364	0360C0	0360C0
P23SSP30	035	18	01	006	1659	036618	039118
P23SPW30	035	19	03	015	0852	038D90	038D90
P23ATP40	036	03	02	001	1641	000000	0005E8
P23SSP40	036	03	03	007	0209	000670	002170
P23SPW40	036	05	02	018	0401	002E98	009AAR
P20ATS	036	09	04	020	0912	000000	006538
*2620AP5	036	14	04	013	0218	000000	000000
*2620EU4	036	18	01	013	0218	000000	000000
*2620J18	037	01	02	013	0218	000000	000000
*2620F8H6	037	04	03	013	0218	000000	000000
*262LEE2	037	07	04	019	0298	000000	000000
*262SJ18	037	15	04	013	0218	000000	000000
*262TFP3	037	19	01	019	0298	000000	000000
*262TJ13	038	03	04	013	0218	000000	000000
*262TRI4	038	07	01	013	0218	000000	000000
P04ATP	038	10	02	006	1659	000000	001800
FC080L	038	11	04	017	0650	007800	008470
FC080L10	038	16	01	017	1428	00A360	00A36A
FC080L12	039	00	02	019	1216	00A360	00A36A
FC080L11	039	05	01	016	0140	00A360	00A8D8
FC080L20	039	09	01	013	1264	00A158	00A168
FC080L22	039	12	02	015	0384	00A158	00B148
FC080L21	039	16	01	016	0404	00A158	00A9D0
FC080L30	040	00	01	007	1540	00A158	00C7A8
FC080L40	040	01	04	023	1583	0090A0	012474
FC080L50	040	07	03	022	0048	0090A0	011A70
FC080L51	040	13	01	021	0852	0090A0	011380
FC080L60	040	18	02	016	1426	0090A0	009DAC
FC080L61	041	02	02	002	0573	0090A0	009D84
FC080L70	041	02	04	026	0392	009288	009294
*26A2J03	041	09	02	001	0002	000000	000000
*2625AP4	041	09	03	013	0219	000000	000000
*J08CTL1A	041	12	04	004	0408	000000	000FAR
*J08CTL0	041	13	04	004	0944	000FAR	000FAR
*J08CTLF	041	14	04	004	0453	000FAR	000FAR
*J08CTLG	041	15	04	004	0744	000FAR	000FAR
*J08CTLJ	041	16	04	004	0406	000FAR	000FAR

root phase
of COBOL
compiler

see page 8-7

supervisor
transient
routine

SYSTEM DIRECTORY

CORE-IMAGE

RELOCATABLE

SOURCE-STATEMENT

24/01/78

-----DECIMAL-----

	C H R E	C H R E	C H R E
DIRECTORY STARTING ADDRESS	00 10 01	46 00 01	61 00 01
DIRECTORY NEXT ENTRY	00 13 11 04	46 01 14 15	61 00 20 09
DIRECTORY LAST ENTRY	00 15 15 17	46 04 17 19	61 01 27 09
LIBRARY STARTING ADDRESS	00 16 01	46 05 01	61 02 01
LIBRARY NEXT AVAILABLE ENTRY	42 11 01	57 06 02	88 14 15
LIBRARY LAST AVAILABLE ENTRY	45 19 04	60 19 16	98 19 27

-----STATUS INFORMATION-----

DIRECTORY ENTRIES ACTIVE	975	609	187
LIBRARY BLOCKS ALLOCATED	3616	4720	20466
LIBRARY BLOCKS ACTIVE	3288	3357	14373
LIBRARY BLOCKS DELETED	52	180	545
LIBRARY BLOCKS AVAILABLE	276	1183	5543
AUTOMATIC CONDENSE LIMIT	00	00	00
LIBRARY ALLOCATED CYLINDERS	45	15	38
DIRECTORY ALLOCATED TRACKS	06	05	02

8-5

CSD.SFW.05

628

545
629

546

CSERV: The purpose of CSERV is to print out on the printer and/or punch on the card punch the members of the core image library.

Format of control card:

Ø operation Ø operand [, operand 2]

Operation: May be one of the following:

DSPLY

PUNCH

DSPCH

Operand: May be one of the following:

phasename

progl.ALL

ALL

EXAMPLE: Print out all of the phases of FCOBOL

// JOB PRTFCOB

// EXEC CSERV

Ø DSPLY FCOB.ALL

/*

/&

CORE IMAGE LIBRARY

PHASE FOURDL30 LENGTH -011,668 BYTES.

00A150 06C3D6C2 D6D3F3F0 F3F687EA D5011002 10064780 87849500 10054770 87E49513
00A180 10061002 474087FA 92009882 D2009883 100317AA 43A10002 89A00006 42A09884 D6009884 10050200
00A1P0 89A00006 42A09886 D6009886 10090201 91A59883 D20191AD 98959201 91A79228 91AF0A09 92F19892
00A1F0 47F087D8 92F39882 47F087D8 92F49882 47F087D8 92F59882 47F087D8 D2009C46 98920203 8C538DCE
00A210 92F08631 96F08631 47F0874A 92F29844 92F08631 47F0874A 92F39844 92F08631 47F0874A 92F49844
00A240 92F59844 92F08631 47F0874A 92F69844 92F08631 47F0874A 4120E0CC 58709798 D2077068 2010D207
00A270 2008D207 71942018 4120E0EC 587097A0 D2077068 2010D207 709C2000 D20770D0 2008D207 71942018
00A2A0 4550890C D2004041 9151D201 40439152 92064020 41108D1E 04009180 10024710 88CA0A07 91401005
00A2D0 F0009180 8D8347F0 88F6D203 91A891A4 58108D02 92FF1000 D23C1001 10004160 E018D73D 10006030
00A300 E01447F0 841041C0 70684180 E0404120 E0814530 894C41C0 709C4180 E0504120 E08F4530 894C41C0
00A330 E0FD4530 894C41C0 71944180 E06C1822 4530894C 07F5D503 C000916C 0783D200 4041C001 00000000
00A360 C0DEC0DE 91000000 0000A2A8 0000A52C 0000A4CC 0000A424 0000A421 0000A380 0C0000AC 00F00078
00A390 0C000000 0C000000 0C000000 0C000000 0C000000 0C000000 0C000000 18211811 D2008D12 0C000000
00A3C0 91581821 18114310 40435C00 91581A21 43004044 06001A20 41100200 19214720 883C0620 5A20907C
00A3F0 95018900 47708A12 92C0F000 92054020 0A009180 00000209 000001FF 003FFF00 00000200 00000200
00A420 000058F1 400000FF 0000A658 003FFFFF FF000000 00FFFFFF D5002000 90002000 00000000 00000000
00A450 C00000FF D2002000 8000D200 20009C00 00000000 50C0F064 58C0F060 90F1C290 5810C150 D207C154
00A480 D203C1F0 11C89101 11404710 F0549201 11405810 114041F0 0006D7FE 10001000 411100FF 46E0F036
00A4B0 C21858F0 C15405FF 98F1C290 58C0F064 07FF0000 0000A164 00000000 9037C22C 90FFC240 18525450
00A4E0 0C095C40 C2285870 C1504177 00AC5877 00005877 00004135 70009180 30004780 F04C9160 30004710
00A510 C22405FF 98FFC240 58030000 5400C2D0 5420C2A4 1A209837 C22C07FE 5090C248 41A00004 18881800
00A540 18901888 5080C2A0 41A88000 1A885880 C1505888 014041AA 8000D202 C28DA000 5880C28C 5890C248
00A570 58C0F218 90FFC290 91081000 4780F078 9048C258 18A69104 10004770 F0405820 C2DC5850 C2E01255
00A5A0 4780F048 41E0000C 47F0F186 58210004 5020C2DC 58F0C20C 05EF58F0 C2949180 20004780 F05F4122
00A5D0 F038D202 C2852000 5850C284 5450C2C8 47F0F088 91041000 4710F1CC 9048C258 5850C2C4 18335891
00A600 58F0C208 05EF58F0 C2941288 4770F04C 41E00004 47F0F186 12664770 F08A5980 C2DC4740 F1281828
00A630 C2941877 91802000 4780F0E2 D202C285 20005870 C2845470 C2C84122 00031266 07761985 4780F11C
00A660 F0FC18A2 41220003 4480C2D4 4770F11C 12334780 F11241F0 000847F0 F1861838 184AD202 C288A000
00A690 47F0F0AC 12334770 F13641E0 000447F0 F1861823 58F0C20C 05EF58F0 C29491C0 20004780 F1564122
00A6C0 00034122 80019110 10004780 F1701244 4770F186 41F0000C 47F0F186 910C1000 4780F19C 12444770
00A6F0 F19FD202 C285C288 5810C284 5410C2C8 5010C2E0 47F0F1A2 18445040 C2E05030 C2DC9848 C25858E0
00A720 07FE50E0 F21C9848 C25858E0 C29058C0 F22058F0 F21C07FE 5090C258 58210000 5420C2D0 5020C2DC
00A750 C29491C0 20004780 F1FA4122 00034740 F1FA4122 00031888 5080C2E0 43820000 41228001 5880C258
00A780 07FE0000 0000A164 00000000 0000881C 50C0F0EC 58C0F0E8 90FFC290 5040C258 182158F0 C20C05EF
00A7B0 5A00C2AC 91C02000 4780F038 41220003 4740F038 41220003 18444342 00004124 20014342 000091F0
00A7E0 20004710 F05A4122 00014122 00064342 00004122 400147F0 F08C959A 20004780 F07A9598 20004770
00A810 5A40C1F8 88400004 41224000 19204780 F09A91FF 20004770 F0825410 C2A84111 02001821 58F0C20C
00A840 F08A1832 5830C284 1A1391C0 20004780 F0CE4122 00034740 F0CF4122 00031832 43420000 41224001
00A870 58C0F0EC 07FE0000 0000A164 40000143 58108C48 05010200 500090DC 47F07014 0000C828 45E08A88
00A8A0 954490E2 478070A8 958190F2 47807060 950590F2 47807050 952290E2 47807050 952490E2 47807050

CSD. SEM. OS

548

RSERV: Performs the print and punch functions for the relocatable library.

Format of control card

␣ operation ␣ operand1 [, operand2]

Operation:

DSPLY

PUNCH

DSPCH

Operand:

Modulename

proq1.ALL

ALL

EXAMPLE: Punch out the printer IOCS module called IJDFAPIZ

// JOB PRTPTR

// EXEC RSERV

␣ PUNCH IJDFAPIZ

/*

/&

IJDFAPIZ	V.M	2.9			00003 BLOCKS	SYSTEM RELOCATABLE LIBRARY
FSD		032	0001		IJDFAPIZ	SD 000000 000118 IJDFAPIZ LD 000000 000001
TXT	000000	136	0001		0A320000 47F0F0A4 0A320000 47F0F01A C9D1C4C6 C1D7C9F9 F3F99180	
					F0240AC7 900FF0FC 58E01018 06E00200 1017E000 18CC180C 43C01017	
					430FF0F7 19DC4780 F05046F0 F0400A32 430EF107 43C01016 19CF4770	
					10164780 F0769208 10280A00 91801002 4710F076 0A074200 10284200	
					91801002 4710F08A	
TXT	000088	136	0001		0AC7D201 10261003 92011028 58E01028 02021029 101950E0 101806F0	
					44001022 58E0F0F4 0A0007FE 91801002 4710F08E 0A079101 10154780	
					10159102 1026078E 47F0F0DA 91011027 078E1200 4780F0F4 18F007FF	
					0A0007FF 00000000 00000000 00000000 F2C2C1F9 F8F7F6F5 F4F3C3F1	
					93C8D3CB 03BB83AB	
TXT	000110	008	0001		A398E388 03181308	
END	404040					

8-9

CSD.SFW.05

634



549

635

SSERV: Performs the print and punch functions for the source statement library. The source statement library is divided into parts called sublibraries. There is one for COBOL called 'C' and one for the assembler called 'A' and in the case of OSAIA one called 'P' for cataloged JCL Books.

Format of control card:

```
% operation % operand1 [ , operand2]
```

Operation;

```
DSPLY
```

```
PUNCH
```

```
DSPCH
```

Operand; sublib.book

```
sublib.ALL
```

```
ALL
```

Example: Print and punch the assembler sublibrary of the source statement library.

```
//JOB SORLIB
```

```
// EXEC SSERV
```

```
% DSPCH A.ALL
```

```
/*
```

```
/&
```

V.M 0.0

18 BLOCKS

PRIVATE SOURCE-STATEMENT LIBRARY

BKEND P.C24B

* \$\$/*

// EXEC LNKEDT

// ASSGN SYS006,X'CRO'

//ASSGN SYS005,X'PRO'

// EXEC

CSM147206

SGM145204

1SG137193

MSG135191

SP7118181

SFC116179

SB6108168

SSG106166

SP5105156

SGT103154

SP4082136

PFC080116

PV207116

PV1067116

* \$\$/*

255656008FOX HENRY E

0101SGM101639264

326366478ISSAACS JUDY I

0101SSG060924487

353120375BERGERT KENNETH B

0101PV1050161948

545508335CROMBAUGH SUSAN C

0101SGM061676629

631608320ESSELBORN RAYMOND

0101CPL080514535

631985218KAZACOFF KOTCHO K

0101SGM051557952

639116773GRIEBELBAUER JOHN

0101SP6001041329

649788627JOHNSON SAM J JR

0101SGT050773699

787425848LEWIS FLOYD L

0101SGC111139387

846775121ADOMATIS RUSSELL

0101E-4130614992

848521418DEZUTTI LOUIS D JR

0101SSG030963552

888124925HOLTZMAN SALLY H

0101PFC040369948

251745252QUINCY WALTER O

0102SO6011054487

2559732500XLEY MORRIS O JR

0102PFC000206662

356879414MCCARTHY MARY M

0102SP6011054487

547083118SMITH PITTMAN S

0102PFC000206662

637427078UNDERWOOD DEVIL H

0102SP6081005936

637474514VERMILLION BETTY V

0102)V2010213992

773137376RIVERA FRANCES R

0102SGT090728487

786214360WHEELER GREGORY W

0102PV1000058436

831195644TOMLINSON HARRY T

0102PV1020303236

849238650NEWTON DWIGHT N

0102SSG031019192

846493958PARKER CLYDE B

0102E-5100832938

248041032XAVER RUTH X

0102PV2040435336

291777069ZUCKERBURG HARRY Z

0102SSG131051363

353865446YATES CLARENCE Y

0103PV1050025653

115678395GRIFFIN ARCHIE G

0103PV1100010593

161312570COSTELLO LOUIS C

0103SP6131040636

337255981DIRKSON EVERETTE D

0201CPL000405288

535973792ABBOTT BUD A JR

0201SSG131051363

569264882FORD GERALD F

0103PV1050025653

737745671HUMPHREY HUBERT H

0103PV1100010593

CSD.SFW.OS

637

552

778712763EAGLETON THOMAS E
825391253ISLEY BROTHERS I
923948125BUTZ EARLE B
263233736JOHNSON LYNDON J
357189154MANSFIELD MIKE M
66588495KISSINGER HENRY K
762042553NIXON RICHARD N
297864442LINCOLN ABRAHAM L
384353898QUEENIE PATRICIA
437293380OVERDRIVE BACHMAN
654619466PATTON GEORGE P
797833538BUMSFELD DONALD R
062535910STEPPENWOLFE CRAZE
297275429TRUMAN HARRY T
196468460UNDERDOG CANINE U
258492158ZIMMERMAN JAMES Z
328339176WALLACE GEORGE W
384422357XAVIER I A
611989829YANKEE CONNETICUT
846244754VICTORIOUS NOT SO

021SGT080624264
0201E-2010364888
0201E-7071187635
0301PV1090206487
0301SFC011117629
03011SG021665787
0301SSG050852753
0301MSG101573938
0302SP5020646935
0302SP6070902498
0302SGT020898236
03020PL160482498
0303SP4010465745
030PFC000361248
0304PV2090315236
0304SSg130989953
0304PV2120025523
0304SGT070745888
0304SP4080448142
0304PV1050331888

8KEND

CATALOG NEW MEMBERS TO LIBRARIES

CORE IMAGE LIBRARY: We have already seen in our discussion of Job Control that the LINKAGE EDITOR is the program that catalogs phases to this library.

EXAMPLE: Catalog an object deck to the core image library and call the new phase CATX.

```
// JOB CATLEX
// OPTION CATAL
Ø PHASE CATX,*
Ø INCLUDE
```

OBJECT DECK

```
/*
// EXEC LNKEDT
/&
```

RELOCATABLE LIBRARY: The program that catalogs modules into this library is called MAINT.

Format of control card:

```
Ø CATALR modulename [ , v.m]
```

Modulename A unique name consisting of 1 to 8 alphameric characters the first of which is alphabetic.

v.m This optional entry is used to provide a version and
 modification number to further identify this module from
 other previous copies with the same name.
 v can range from 0 to 127.
 m can range from 0 to 255.

EXAMPLE: Catalog an available object module to the relocatable
 library and call it MOD516. For this copy use this
 month (June) for the version and today's date (27th) for
 the modification number.

```
// JOE CATALREX
// EXEC MAINT
0 CATALR MOD516, 6.27
```

OBJECT MODULE

```
/*
```

```
/&
```

SOURCE STATEMENT LIBRARY: The program that catalogs to this library is called MAINT.

Format of control cards:

MCATALS	Sublib.bookname [,v.m. [, C]]
Sublib	The single alphabetic character which identifies which sublibrary this book is to be cataloged into.
bookname	The unique name for this book within its sublibrary. 1-8 alphameric character the first of which must be alphabetic.
v.m.	Further identification to allow us to differentiate this from previous versions.
C	This character is entered as part of the directory and, if used, requires the user to specify the correct version and modification number if he is trying to change the module using the UPDATE feature of MAINT.

⌘ BKEND [sub.book] [,SEQNCE] [,count] [,CMPRSD]

This control card must appear in front and behind of each book that is cataloged into the source statement library. The optional operand entries, if used, need only be put in the front BKEND card.

Sub.book	same as operand in CATALS card
SEQNCE	The sequence numbers punched in card-columns 76 to 80 are checked for ascending sequence numbers.
Count	used to specify the number of cards in the book including the two BKENDs.
CMPRSD	compress card formats by removing blanks.

EXAMPLE: Catalog into the source statement library a standard file description book and call it STDFD74.

```
// JOB CATALGS
// EXEC MAINT
⌘ CATALS C.STDFD74
⌘ BKEND C.STDFD74 ,,,CMPRSD
```

SOURCE BOOK

⌘ BKEND

/*

/&

042

RENAMING MEMBERS OF LIBRARIES

The program MAINT performs this function for all three libraries.

Core Image Library

```
// JOB EXAMCIL
// EXEC MAINT
  RENAME oldphasename, newphasename
/*
/
```

Relocatable library

```
// JOB EXAMRLL
// EXEC MAINT
  RENAME oldmodulename, newmodulename
/*
/
```

Source Statement library

```
// JOB EXAMSSL
// EXEC MAINT
  RENAME sublib.oldbookname, sublib.newbookname
/*
/
```

NOTE: All of these operations could have been done in a single job.

2. The operand field can be repeated to specify more than one rename operation.

DELETING MEMBERS FROM LIBRARIES

The program MAINT performs this function for all three libraries.

CORE image library

```
        // JOB EXDELCIL
        // EXEC MAINT
        Ø DELETC phasename
or      Ø DELETC prog.ALL
        /*
        /&
```

NOTE: prog.ALL will delete every phase in the core image library whose name begins with the 4 characters specified.

Relocatable library

```
        // JOB EXDELRLI
        // EXEC MAINT
        Ø DELETR modulename
or      Ø DELETR Prog.ALL
or      Ø DELETR ALL
        /*
        /&
```

NOTE: Prog.ALL deletes all modules whose first 3 characters are specified.
ALL deletes the entire library.

SOURCE STATEMENT LIBRARY

```
        // JOB EXDELSSL
        // EXEC MAINT
        ⚡ DELETS Sublib.bookname
or      ⚡ DELETS sublib.ALL
or      ⚡ DELETS ALL
        /*
        /&
```

NOTE: Sublib.ALL deletes the entire sub-library
ALL deletes the entire library.

CONDENSING LIBRARIES

If a delete is performed the reference to the member in the library directory is removed. However, the old copy still occupies space in the library itself.

The same is true if a new member is cataloged with the same name as an old member. The old member is deleted from the directory but still occupies space in the library itself.

The program MAINT perform the condense function for all three libraries.

```
// JOB EXCONDLB
// EXEC MAINT
Ø CONDS CL
or Ø CONDS RL
or Ø CONDS SL
or Ø CONDS CL, RL, SC
/*
/&
```

UPDATE (Change) SOURCE LIBRARY BOOKS

Once books are cataloged to the source statement library there is no need to maintain them in punched card format. However, there may be a requirement to make corrections or minor changes to library members. One way to do this is to punch up a new copy with changes and catalog it over the old copy. Another method would be to use the UPDATE function of the program MAINT to make these minor changes.

NOTE: In order to update a book it must have been sequenced prior to being cataloged. In the book card columns 73 - 76 will contain the 1st four characters of the bookname and card columns 77 - 80 will contain ascending sequence numbers.

The control cards are as follows:

```
// EXEC MAINT
% UPDATE sublib.bookname, [s.book], [v.m], [nn]
) ADD seq-no
) REP first-seq-no [ ,last-seq no]
) DEL first-seq-no[ , last-seq-no]
) END [v.m.[,C]]
```

UPDATE sublib.bookname ,[s.book] ,[v.m.] , [mm]

Sublib.bookname The name of book being updated

s.book The temporary name given to the old book to preclude deleting it, in the event that an error was made in the update.

NOTE: If the update was successful this temporary book would have to be deleted.

v.m. An optional that would be required if the old book requires change level verification.

mm. This is the interval to be used when new book is assigned sequence numbers. Default is 01 if no resequence desired enter "NO" in this position.

) ADD seq-no

Seq-No applies to the sequence number in card columns
77 - 80 of member book cards.

ACTION: The 'seq-no' in the ADD card serves to indicate behind which card in the oldbook to place the cards that follow.

EXAMPLE:

OLD BOOK

05 Filler	Pic X (10)	SDFD0120
	Value spaces.	SDFD0130
05 Filler	Pic X (9)	SDFD0140
	Value 'LAST NAME'.	SDFD0150

CONTROL CARDS

```
// JOB UPDATEIT
// EXEC MAINT

UPDATE C.SDFD150,,,10

) ADD 0130

    05 Filler          Pic X(4)

                        Value 'Dept'.

) END
/*
/&
```

NEW BOOK

05 Filler	Pic X(10)	SDFD0120
	Value spaces.	SDFD0130
05 Filler	Pic X(4)	SDFD0140
	Value 'Dept'.	SDFD0150
05 Filler	Pic X(9)	SDFD0160

564

) REP first-seq-no [, last-seq-no]

ACTION: The cards in the old book starting with 'first-seq-no' and ending with 'last-seq-no' are deleted and the source card(s) that follow the) REP card are put in their place. One or more old cards can be replaced by one or more new cards.

) DEL first-seq-no [, last seq-no]

ACTION: If only the first operand is used, only that card is deleted. If the two operands are used the series of cards beginning with the first and ending with the second will be deleted.

) End [v.m. [,C]]

This card indicates the end of updates to a book.

v.m new version and modification

number to be placed on new book. Suggestion: month.day (6.29)

C indicates change level verification is required before any subsequent updates to this book.

600

REALLOCATION FUNCTION

The reallocate function is used when it becomes necessary to change the size of the libraries on the system resident pack.

```
// JOB CHNGIT
// EXEC MAINT
// ALLOC id=cylin (track) [, id=cylin (track),....
/*
/&
id  this tells the system which library CI (Core image), RL(relocatable),
    or SL (Source statement).
```

cylin Specifies the number of cylinders to be allocated for a particular library.

track Specifies the number of track to be taken from the beginning of the first cylinder to store the library index.

NOTE: for the computation of required space see IBM system control and service manual.

COPY FUNCTION

The copy function perform the following operations, individually or in combination:

1. Defines and/or creates a new system Pack.
2. Defines and/or creates private libraries.
3. Transfers phases, modules, or books between any assigned libraries.

Define and create new system pack

```
// JOB COPY
// ASSGN SYS002, X' 191"
// LABEL IJSYSRS, 'DOS SYSTEM RESIDENCE FILE', 99/365,SD
// EXTENT SYS002,111111, 1, 000,00001, 1219
// EXEC CORGZ
/* ALLOC CL=60(10), RL=30(10), SL=30(10)
/* COPY ALL
/*
/*
```

CORGZ The program phase that performs the copy function.

ALLOC Same as described in Reallocate function.

COPY ALL Cause all of libraries to be copied onto the new pack.

652

EXERCISES

1. Write the job control language necessary to print the directories of the core image library and the source statement library.

2. Write the job control language necessary to cause the following to be printed.

0	P.FDMS57	V.M	1.5	2 BLOCKS	SOURCE STATEMENT LIBRARY	0
0						C
0	BKEND	C.FDMS57				0
0						0
0	00001	RECORD CONTAINS 80 CHARACTERS				0
0	00002	LABEL RECORDS ARE OMITTED				0
0	00003	DATA RECORD IS IT01-CARD-RCD.				0
0	00004 01	IT01-CARD-RCD.				0
0	00005	05 IT01-NAME		PIC	X(20).	0
0	00006	05 IT01-SSAN		PIC	9(9).	0
0	00007	05 IT01-GRADE		PIC	X(2).	0
0	00008	05 IT01-ADDR		PIC	X(20).	0
0	00009	05 IT01-PT-SCORE		PIC	9(3).	0
0	00010	05 IT01-PHONE		PIC	9(4).	0
0	00011	05 IT01-UNIT		PIC	X(10).	0
0	00012	05 FILLER		PIC	X(12).	0
0						0
0	BKEND					0
0						0

568

3. Write the job control language necessary to catalog an object module in card deck form into the proper library and call it ZOOM5.

4. Write the job control language necessary to catalog a set of JCL cards into the proper library and call it PROC97.

5. Write the job control language necessary to delete GOPHASE from the core image library and MOD84 from the relocatable library and also condense all three libraries.

DOS UTILITY PROGRAMS

1. INTRODUCTION. Whatever the specific uses of a data processing system, certain unique operations exist that must be performed frequently. These operations may differ in detail, depending on the particular machine configuration and data format for the individual user, but the essential function remains the same. Generalized routines designed to satisfy specific functions would ease the burden of programming these operations. These routines must be flexible enough to allow the user to assign the specifications of his particular problem.

One type of program that meets these requirements are utility programs. They are designed to assist the user in day-to-day operation of his installation. With these programs, the user can perform certain frequently required operations, such as transferring disk storage files to cards or tape and printing out areas of tape or disk for program testing purposes, without programming effort.

2. EXPLANATIONS OF THE UTILITY SPECIFICATIONS.

a. File to File Utilities. Eleven utility programs are provided for the transfer of data files from any of the normal input devices to any of the normal output devices. A file can be transferred between unlike storage mediums (tape to disk), like mediums (tape to tape), or, in the case of disk to disk, the files may be transferred from one area to another area of the same unit.

A file can be transferred from one input medium to an output medium with these option:

- * COPY - indicates that the file is to be transferred from an input medium without change to the format of the records or file.
- * REBLOCK - transfers the input file from an input medium to an output medium with only the block size being changed.
- * FIELD SELECT - rearranges or drops fields within each input record, or converts them to zoned or packed decimal.
- * REBLOCK AND FIELD SELECT - is a combination of the reblock and field select options.

Printer output allows you to show the output in three ways:

- * DISPLAY - allows you to display a byte for byte representation of the information.
- * LIST - gives an edited representation of the information.
- * LIST AND FIELD SELECT - is a combination of the list and field select options.

For the card to printer and/or punch programs, two other output options are:

- * BOTH PRINT AND PUNCH - is a combination of copy and list.
- * BOTH PRINT AND PUNCH WITH FIELD SELECT - combines copy and list with field select.

These programs will handle fixed-length, variable-length, and undefined-length records, however, only fixed or variable length records can be reblocked or field selected.

b. Label Checking. Tapes containing standard labels or no labels at all can be processed without providing a user routine. When using a file to file utility program to process nonstandard or user labels or when no label checking is desired, an UPSI job control card is required. Only the 0 and 2 bits are normally used. When bit 0 is OFF, standard labels are checked on input and when it is ON, nonstandard or no label input is checked. Bit 2 OFF indicates output standard labels while bit 2 ON indicates nonstandard are not labeled.

When an UPSI card is supplied to a program, the byte is propagated from job step to job step unless another UPSI card is supplied to reset the bits. All of the UPSI bits are set to 0 following each job performed unless a new statement is supplied. When right most bits are not set by an UPSI statement, they are assumed to be zero.

c. Multifile Volume (tape). The utility programs may be used to build-multifile volumes and read from them at later dates. File positioning will be performed by logical IOCS only if the files are labeled with standard labels. Output files, nonstandard labeled files and unlabeled files are not acceptable. The file-name, volume-sequence, and file-sequence numbers must be placed in the T.B.L card. When using the utility programs to process multifile tape input volumes, the non-rewind-option (IN) parameter found in the utility modifier statement must be specified.

d. Multivolume Files (tape). Input or output files to these programs can consist of multiple volumes and must belong to the same data files. The control statement entries used to process the first volume are used to process each successive volume because the same fields are checked in each volume. Each tape reel of a multivolume tape file is unconditionally rewound and unloaded if no alternate tape drive has been assigned. In all other cases, the volume will be treated as specified by the input or output parameter in the utility modifier statement.

When alternate tape drives are specified and processing is completed on a particular file, the last drive processed will become the primary drive. If a new job is executed at this time, the last drive processed will then become the primary drive unless a reassignment of tape drives is made.

e. Record Skipping. Any number of logical records (up to 99,999,999) may be bypassed before processing is performed. This number can be indicated in the Rx parameter of the utility modifier statement. The number indicated

in the parameter will be the first record to be processed. Record skipping cannot be performed for the copy function (TC), and if specified, it will be ignored. To skip records at the beginning of a file and copy the remainder, the reblock function (TR) must be indicated, and the input-description and output-description parameters must contain identical values.

f. Sequence Numbering. Sequence generation on card output can be indicated in the utility modifier statement. A field up to ten characters long can be punched into each card. This field is numbered starting from 1 (with high-order zeros) and is increased by 1 for each succeeding card. If a sufficiently long field is not defined to number all of the cards, the number wraps around to zero without an error indication. The sequence number overlays any data selected into the sequence area of the card. Sequence checking can also be performed for card input to assure ascending sequence of the specified field. If a card is out of sequence, a message is written on SYSLST and processing continues.

g. Binary Records. When processing cards punched in column binary format, the input and output parameters (A and B in the utility modifier statement) must be specified as twice the number of card columns used.

h. Printer Output. Printer output can be 120, 132, or 144 characters per line depending on the printer used and the output format. Printer output can be data display or list format.

* DATA DISPLAY - this format provides a visual picture of the data file. Fixed, variable, and undefined records can be handled, but the field select option cannot be used. Every byte of data in the file appears in the printed output. If data display is specified (TD), 120-character line is forced. Only portions of the print line are used for data. This first twenty (1-20) are reserved for information describing the file, such as block size, block number, and record number. Data is normally displayed in hexadecimal form, but may be optionally displayed in alphabetic form. A heading line can be printed. A scale line is printed at the top and bottom of each page. If record length is specified as fixed or variable length, each logical record starts on a new line. The input block size is printed only if the input length is not equal to the specified block size. The excess is not printed when the maximum length block is exceeded. Single spacing is used between lines of print.

* DATA LIST - the data list format provides a simple edited listing of the file. The entire print line is available for data output; output is restricted to one line per logical record. The input record length cannot exceed the size of the print line. If so, the field select is required. Fields can be selected to, unpack, convert to hexadecimal, and format the page. Data list mode allows only character printing unless a hexadecimal field is selected through a field select entry.

i. Utility Modifier Statement. This statement is used to describe the input file that is to be processed and the output file that is desired. If

the statement is present and optional parameters are left out, default values are assumed.

The general format of the Utility Modifier Statement is:

```
// Uxx Tt,Ff,A='input'),B=(output),E=(c), Ix,Ox,Px,Q=(x,y),Rx,Sx
```

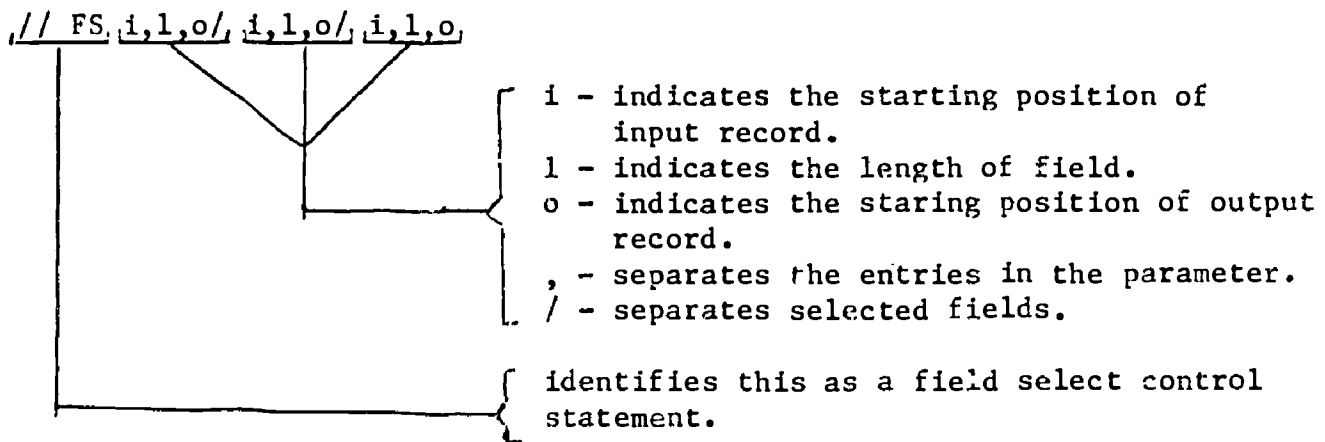
j. Field Select Statement. With this option, a field in each input record can be moved to a different location in the corresponding output record. Those areas of the output record that are not filled with selected fields are filled with blanks.

When field select is used, only those bytes in the input record that are selected will be transferred to the output record. It is therefore possible with field select to ignore certain fields and have them dropped from the output record.

The field select control statement provides the information for the file-to-file programs to transfer fields from an input record to the same or a different relative location of the output record. Each card should begin with //FS%. As many field select statements as necessary may be used and must be complete on one statement. For example:

```
// FS 1,15,1/16,(P,5,3),16/72,5,19
// FS 21,2,30
```

The format and contents of the field select statement are:



k. Heading Statement. A heading line can be printed for programs with printed output. Heading lines are ignored if first-character forms control is specified. A maximum of two statements can be used to indicate the heading line desired, but the second statement need not be entered if the first statement contains all of the desired information. The first statement is entered // H1% (followed by the information to be printed in print positions 1-74. The second heading statement is // H2% (followed by the remainder of the heading line). Heading statements must follow any field select statements used.

1. End Statement. This statement must be the last of the utility control statements in the program and is punched // END.

3. FILE TO FILE UTILITY PROGRAMS:

a. Card to Disk. The card to disk program transfers the contents of a card file from cards to an area of disk. The cards may be punched in EBCDIC or binary. The input records must be fixed-length unblocked, and each logical record must fit on one card. The maximum size input record is 80-bytes or 160-bytes for binary. These files may be copied, reblocked, field selected, or reblocked and field selected.

b. Card to Printer and/or Punch. Input records to this program must be fixed-length and unblocked. Card input and output can be either EBCDIC or binary except for both, printing and punching when it must be EBCDIC. The maximum record size is 80-bytes for EBCDIC and 160-bytes for binary.

(1) Card to Printer. The card to printer program can produce printed output in two formats: Display and List. Sequence checking is performed on the input.

(a) Display. This option transfers to contents to a card file to a printer with each record being placed on one print line. The field select option cannot be performed with display. In this format, the first 20 positions of the print line are reserved for information describing the file. When hexadecimal printout is called for, the entire card is printed on two lines.

(b) List. The input records for this option are transferred to the printer with each record being fully printed. The field select option may be used. The full print line is available for printing. When hexadecimal printout is called for, the output record size is bound by the size of the print line.

(2) Card to Punch. Records may be copied or field selected. Sequence fields are generated, but input is not checked.

(3) Card to Printer and Punch. This program produces printed output in the list format and a reproduced card deck. Sequence fields are generated, but input is not checked.

c. Card to Tape. The card to tape program transfers the contents of a card file to tape. The cards may be punched in EBCDIC or binary. The input records must be fixed-length unblocked and each logical record must fit on one card. The maximum record size is 80-bytes for EBCDIC or 160-bytes for binary.

d. Disk to Card. The disk to card program transfers the contents of a disk file to a card file. The output file may be punched in either EBCDIC or binary. Each logical output record must fit on one card. Input records to this program must be fixed-length and may be blocked. Files in this program may be copied, reblocked, field selected, or reblocked and field selected.

All blocked input records must be reblocked.

e. Disk to Disk. The disk to disk program transfers a file between disk units or between areas of the same unit. Using the same device for input and output can cause a reduction in performance. Files can be copied, reblocked, field selected, or reblocked and field selected. If the field select or reblock options are used, the input records must be fixed or variable length.

f. Disk to Printer. The disk to printer program can display a disk file in either data display or data list format. Data display provides a visual picture of the data where every byte appears in the printed output; this format can handle fixed, variable, and undefined records. Data list provides a simple edited list of the file; input records must be fixed or variable length and (for the list function only) must not exceed the size of the print line. An option is available to this program to specify the number of logical records in a file to be bypassed before printing begins.

g. Disk to Tape. The disk to tape program transfers a file from one or more disk units to one or more tape reels. These files may be copied, reblocked, field selected, or reblocked and field selected. If the reblock or field select options are used, the input records must be fixed or variable length.

h. Tape to Card. This program transfers a tape file to a card file. The output file may be punched in either EBCDIC or binary. Input records must be fixed length. The files may be copied, reblocked, field selected, or reblocked and field selected. Blocked input records must be reblocked.

i. Tape to Disk. The tape to disk program transfers a file from one or more tape reels to a maximum of 'N' disk units where 'N' is the number of disk units assigned. These files may be copied, field selected, reblocked, or reblocked and field selected. If the field select or reblock options are used, the input records must be fixed or variable length.

j. Tape to Printer. The tape to printer program can display a tape file in either data display or data list format. Data display provides a byte-for-byte representation of the data file where every byte appears in the listing; this format can handle fixed, variable and undefined records. Data list provides a simple edited representation of the file; input records must be fixed or variable length and (for the data list function only) must not exceed the size of the print line. The field select option may be used. An option is available to this program to specify the number of logical records in a file to be bypassed before printing begins.

k. Tape to Tape. The tape to tape program transfers a file from one or more reels to tape to one or more other reels. These files may be copied, field selected, reblocked, or reblocked and field selected. Input records must be fixed or variable length if the reblock or field select options are used.

4. PHASE NAMES FOR UTILITY PROGRAMS. The phase names listed below are for the utility programs described above.

<u>PHASE NAME</u>	<u>PROGRAM DESCRIPTION</u>
CDDK	Card to disk
CDPP	Card to printer and/or punch
CDTP	Card to tape
DKCD	Disk to card
DKDK	Disk to disk
DKPR	Disk to printer
DKTP	Disk to tape
TPCD	Tape to card
TPDK	Tape to disk
TPPR	Tape to printer
TPTP	Tape to tape

5. UTILITY MODIFIER STATEMENT PARAMETER. The contents of the utility modifier statement parameter is described below.

FUNCTION

TB - Both print and punch
 TBF - Both print and punch with field select
 TC - Copy
 TD - Display
 TF - Field Select
 TL - List
 TLF - List and field select
 TR - Reblock
 TRF - Reblock and field select

FORMAT

FF - Fixed length
 FV - Variable length
 FU - Undefined length

INPUT DESCRIPTION

A=(n,m) where 'n' is record length and 'm' is block length

OUTPUT DESCRIPTION

B=(n,m) where 'n' is record length and 'm' is block length
 B=(p) where 'p' is normally 132 for a 1403N1 printer

INPUT MODE

I1 - EBCDIC
 I2 - Binary
 IN - Do not rewind before or after data transfer (tape only)

INPUT MODE - continued

IR - Rewind before and after data transfer (tape only)

IU - Rewind before and after and unload (tape only)

OUTPUT MODE

O1 - EBCDIC (punch only)

O2 - Binary (punch only)

ON - Do not write disk check (disk only) or do not rewind before or after data transfer (tape only)

OC - Character (printer only)

OR - Rewind before and after data transfer (tape only)

OU - Rewind before and rewind and unload after data transfer (tape only)

OX - Hexadecimal (printer only)

OY - Write disk check (disk only)

SEQUENCE NUMBERING

Q=(x,y) where 'x' is first position of a field for sequence numbering (1 or 2 digits) and 'y' is length of field not to exceed 10 positions

PAGE NUMBERING

PN - Do not number pages

PY - Number pages

FIRST RECORD

Rx - where 'x' is the first record to be processed, normally 1

SPACING

S1 - Single space or stacker select pocket 1

S2 - Double space or stacker select pocket 2

S3 - Triple space or first character stacker control

SA - These

SB - codes are

SC - not normally

SC - used

6. COMMONLY USED UTILITY MODIFIER STATEMENT PARAMETERS. The list shown below are the most frequently used utility modifier parameters.

// EXEC CDDK (card to disk)

// UCD TR,FF,A=(80,80),B=(80,1680),E=(2314),R1,OY

// EXEC CDPP (card to card)

// UCP TC,FF,A=(80,80),B=(80,80),I1,O1,R1,S1

COMMONLY USED UTILITY MODIFIER STATEMENT PARAMETERS - continued

```
// EXEC CDPP    (card to print double space)
// UCP TL,FF,A=(80,80),B=(132),I1,OC,PY,S2,R1

// EXEC CDPP    (card to card and print single space)
// UCP TB,FF,A=(80,80),B=(80,80),I1,OC,PY,R1,S1

// EXEC CDTT    (card to tape)
// UCT TR,FF,A=(80,80),B=(80,1600),I1,OR,R1

// EXEC DKCD    (disk to card)
// UDC TR,FF,A=(80,1680),B=(80,80),R1,S1,E=(2314)

// EXEC DKDK    (disk to disk)
// UDD TR,FF,A=(80,1680),B=(80,1680),E=(2314,2314),OY,R1

// EXEC DKPR    (disk to print double space)
// UDP TL,FF,A=(80,1680),B=(132),E=(2314),PY,OC,S2,R1

// EXEC DKTP    (disk to tape)
// UDT TR,FF,A=(80,1680),B=(80,1600),E=(2314),R1,OR

// EXEC TPCD    (tape to card)
// UTC TR,FF,A=(80,1600),B=(80,80),IU,O1,R1,S1

// EXEC TPDK    (tape to disk)
// UTD TR,FF,A=(80,1600),B=(80,1680),E=(2314),IR,OY,R1

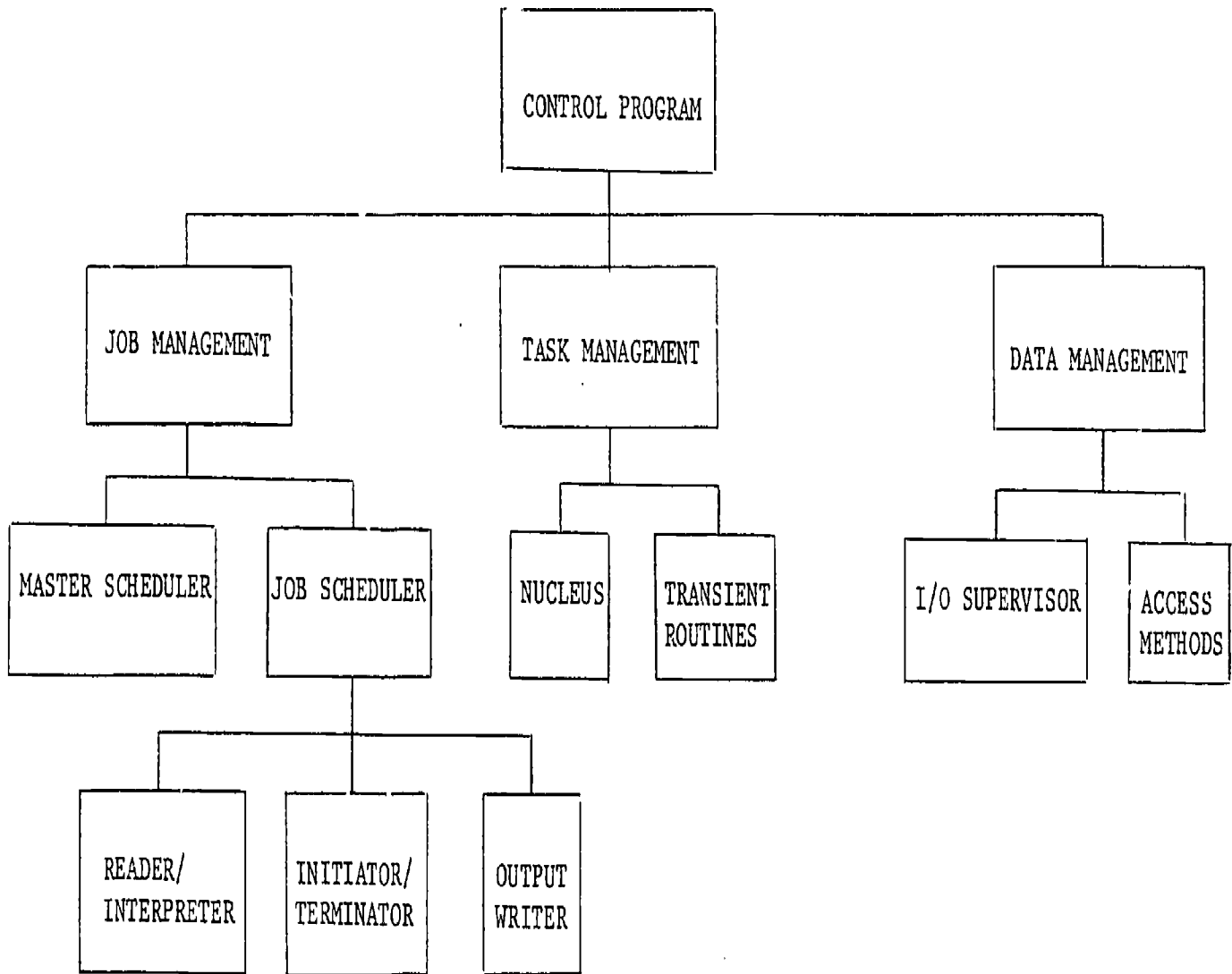
// EXEC TPPR    (tape to print double space)
// UTP TL,FF,A=(80,1600),B=(132),IU,PY,R1,S2

// EXEC TPTP    (tape to tape with field select)
// UTT TRF,FF,A=(80,80),B=(81,81),IU,OU,R1
```

7. For additional information on the utility programs, refer to IBM SRL GC 24-3465, DOS and TOS Utility Programs.

THE OPERATING SYSTEM (OS)

578



10-1

CSD.SFW.OS

665

```
//C23      PROC DUMP='DUMMY'

/*      THIS PE READS NAMES AND LISTS THEM

/*      CREATED ON 30 NOV 76 BY SOFTWARE DIV, CSD, USAIA

//COB      EXEC PGM=IKFCBL00

//      PARM='LOAD,SUPMAP,SIZE=84K,BUF=8K,NODECK,CLIST'

//STEPLIB  DD DSN=USAIA.SOFTWARE.LINKLIB,DISP=SHR

//SYSPRINT DD SYSOUT=A

//SYSPUNCH DD DUMMY

//SYSLIB   DD DSN=USAIA.SOFTWARE.SOURCE,DISP=SHR

//SYSUT1   DD UNIT=SYSDA,SPACE=(460,(700,100))

//SYSUT2   DD UNIT=SYSDA,SPACE=(460,(700,100))

//SYSUT3   DD UNIT=SYSDA,SPACE=(460,(700,100))

//SYSUT4   DD UNIT=SYSDA,SPACE=(460,(700,100))

//SYSLIN   DD DSN=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,

//      SPACE=(80(500,100))

----- COBOL source deck would be placed here -----

//LKED     EXEC PGM=IEWL,PARM='LIST,XREF,MAP,LET',COND=(8,LT,COB)

//STEPLIB  DD DSN=USAIA.SOFTWARE.LINKLIB,DISP=SHR

//SYSLIN   DD DSN=&LOADSET,DISP=(OLD,DELETE)

//SYSLMOD  DD DSN=&GOSET(COB),DISP=(NEW,PASS),UNIT=SYSDA,

//      SPACE=(1024,(50,20,1))

//SYSLIB   DD DSN=SYS1.CORLIB,DISP=SHR

//SYSUTI   DD UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(1024,(50,20))
```

580

```
//SYSPRINT DD SYSOUT=A  
  
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((8,LT,COB),(4,LT,LKED))  
  
//SYSOUT DD SYSOUT=A  
  
//SYS005 DD SYSOUT=A  
  
//SYS006 DD DSN=USAIA.SOFTWARE.DATASET.C09,DISP=SHR
```


OS JCL to execute a COBOL PE

```
//name      JOB (procname,room),'programername',MSGLEVEL=(1,1)
//          EXEC procname
//COB.SYSIN  DD  *
```

COBOL
source
deck

```
/*
```

```
//
```

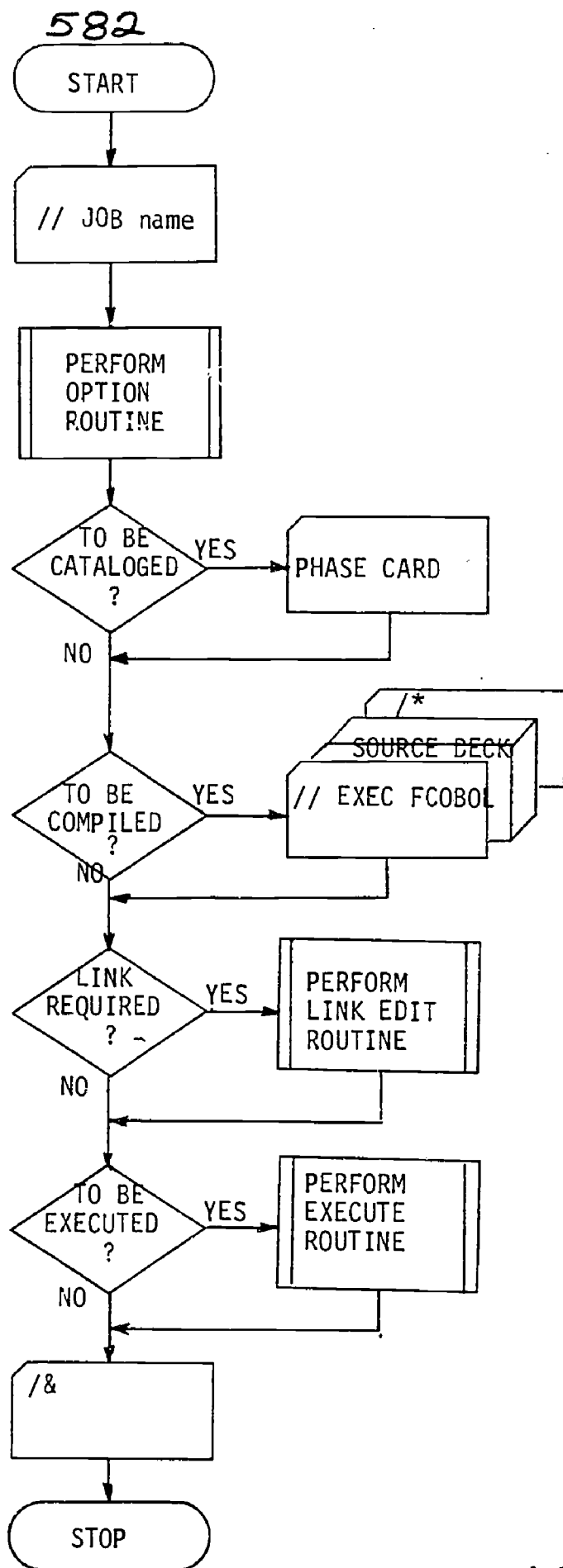
name This is a name assigned to the job. This parameter is taken from the 1st eight characters of your name on the 'buff colored' control card you now put on your deck.

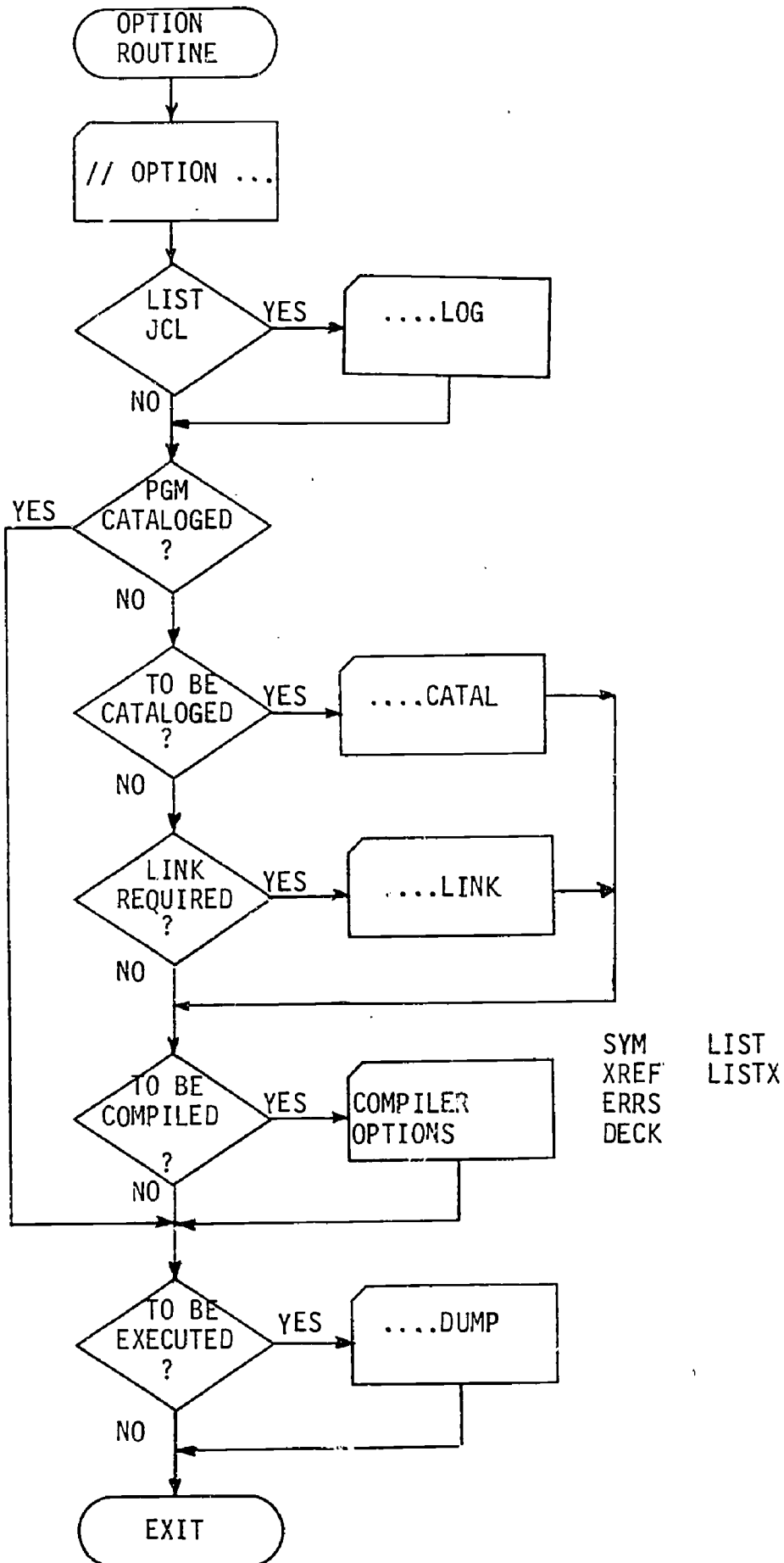
procname Use the code from the buff colored card C11,C09,C23,....

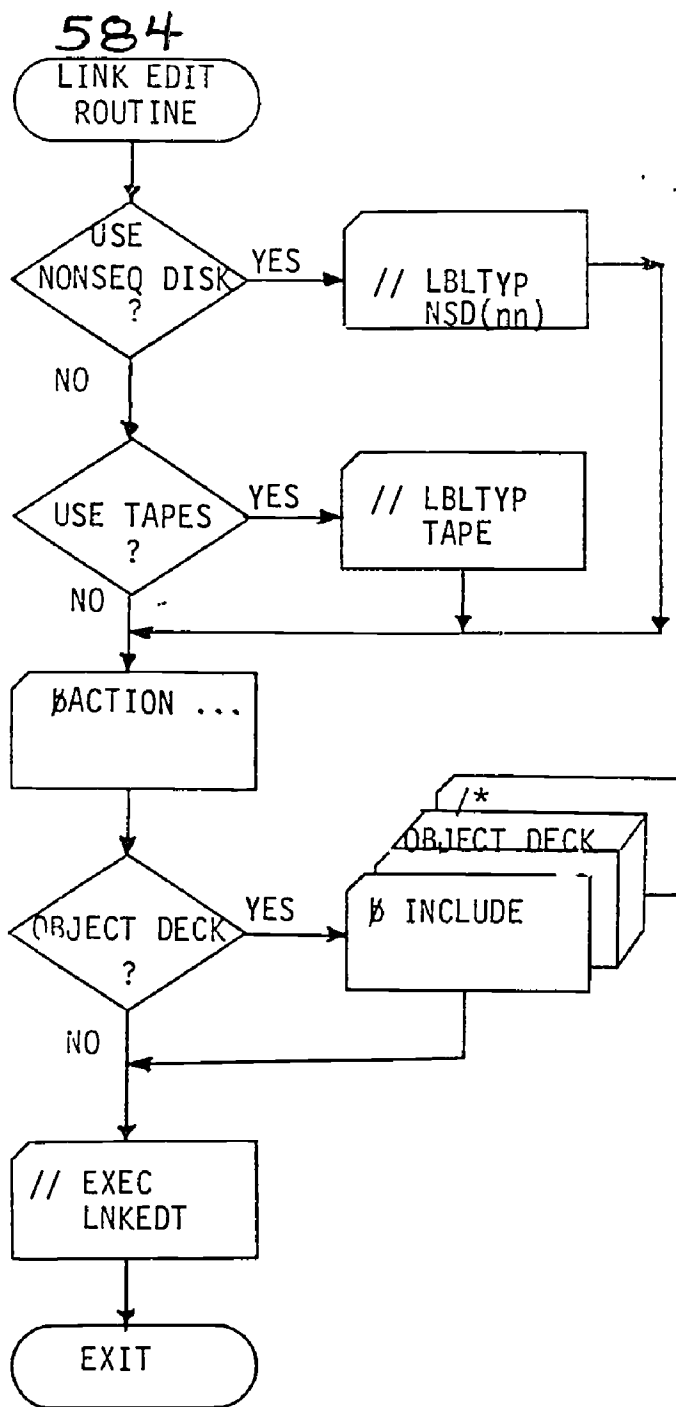
room Enter your classroom number

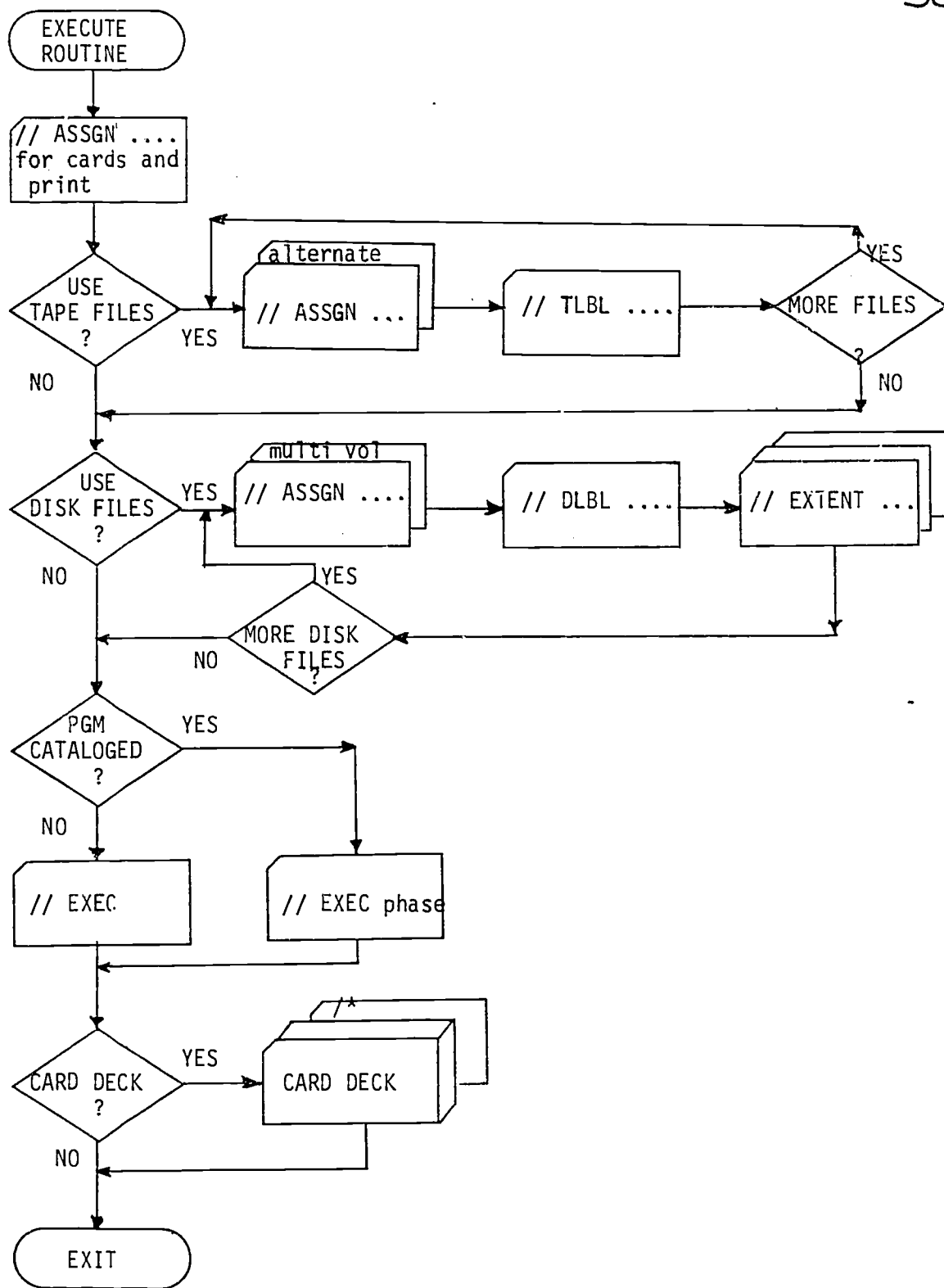
programername 'John Q. Smith'

procname Same as above.









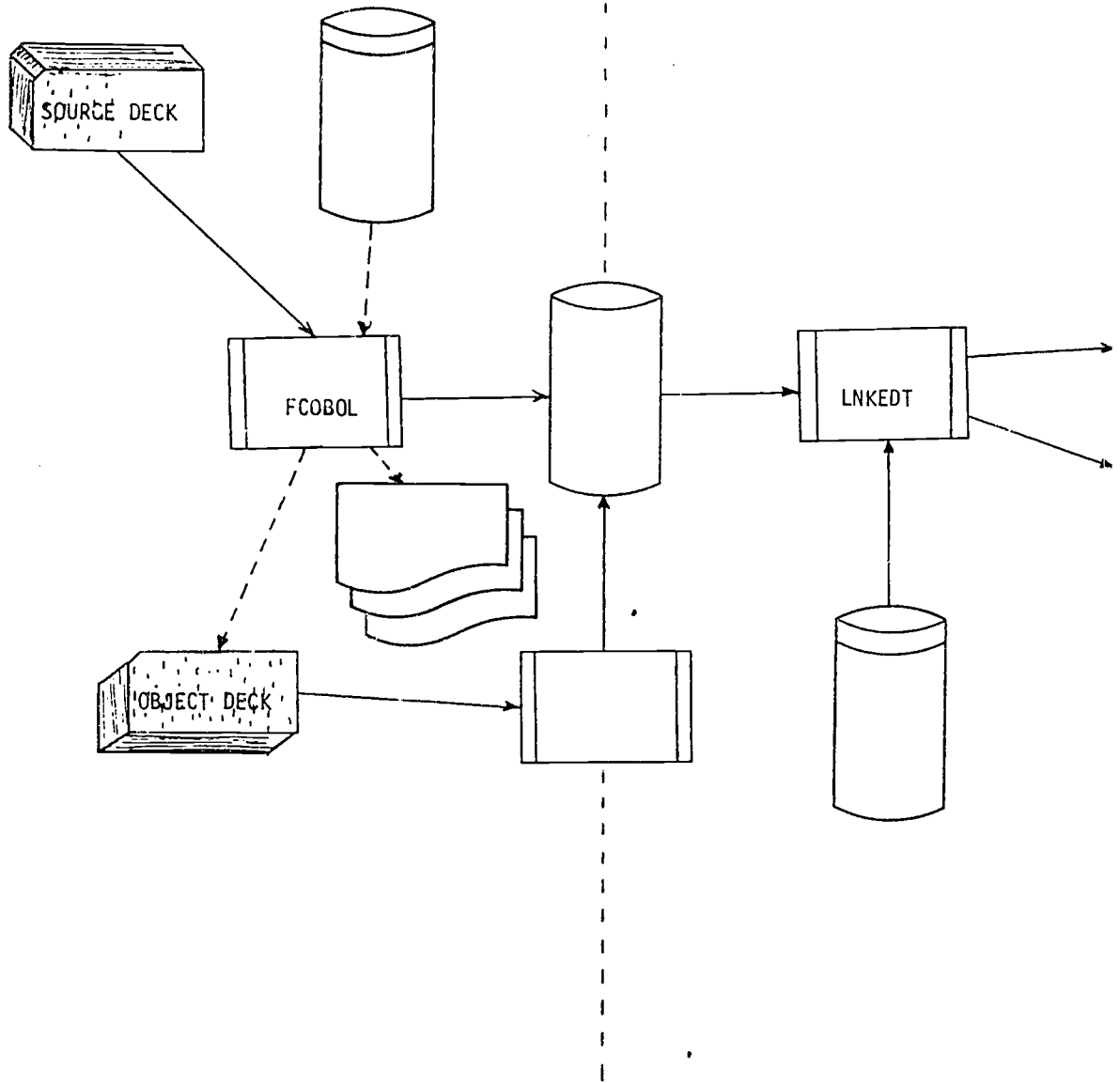
USAFPP 92-51 4/79

8-1

CSD-SFW-05

COMPILE

LINK EDIT



IA-01-01-05 (D)

SYS/360 OPERATING CONCEPTS

OBJECTIVES:

1. To describe in detail the three mainframe components of the IBM SYS/360 computer, Main Memory, the Central Processing Unit, and Data Channels.
 2. To describe the five machine language formats.
 3. To describe interrupts and interrupt processing.
 4. To define PSW, IOCS, LIOCS, and PIOCS.
- I. MAIN MEMORY.- Where data and instructions must ultimately be to be operated upon by the system.
- A. Byte -
 - B. Addressing -
 - C. Concept of "Word Units"
 1. Fullword -
 2. Halfword -
 3. Doubleword -
 4. Boundary Alignment

II. CENTRAL PROCESSING UNIT (CPU).

- A. Instruction Cycle.
 1. I-Time -
 2. D-Time -
 3. The Difference Between Instructions and Data -

IA-01-01-05

- B. Access Width -
- C. Access Time -
- D. Control Element - Fetches and decode machine instructions and sets up internal logic paths for the actual execution of the instruction.
- E. Arithmetic and Logical Unit (ALU) - Responsible for actually executing the instructions as decoded by the Control Element.
 - 1. General Purpose Registers (GPR).
 - a. Size and Number -
 - b. Data Format -
 - c. Uses
 - 1)
 - 2)
 - 2. Floating Point Registers (FP REG).
 - a. Size and Number -
 - b. Data Format -
 - c. Use
 - 3. Operations.
 - a. Arithmetic -
 - b. Logical -

III. DATA CHANNELS - Special purpose minicomputers whose only function is to monitor I/O operations.

A. Modes of Operation.

1. Burst -

2. Multiplex -

B. Types of Channels.

1. Selector -

2. Multiplexor -

C. Channel Command Word (CCW).

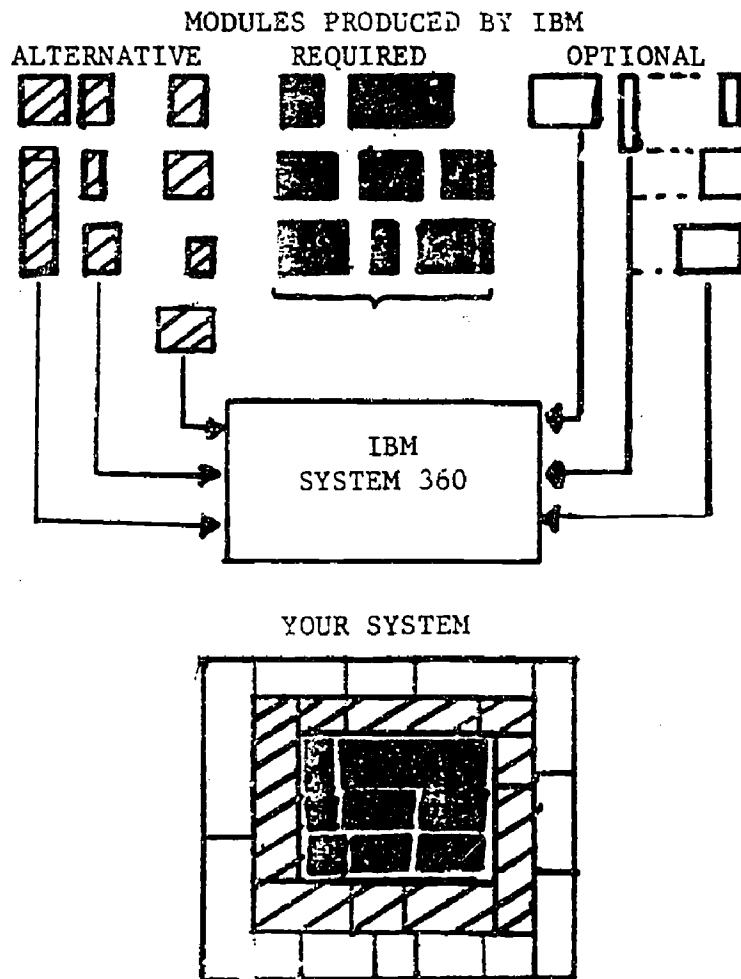
D. Characteristics of Channels.

1.

2.

3.

IV. THE MODULAR CONCEPT OF THE SYS/360.



V. MACHINE LANGUAGE INSTRUCTION FORMATS.

A. The Two Theories of Machine Language Instruction Formats.

1. Fixed Length

- a.
- b.
- c.

2. Variable Length

- a.
- b.
- c.

678

B. Parts of a Machine Language Instruction.

1. Operation Code (OPCODE) - The first byte of the instruction is always the OPCODE. It defines the arithmetic or logical or I/O operation that is to be done.
2. Operands.
 - a. Registers -
 - b. Storage -
 - c. Data -

C. Formats.

1. Register to Register (RR).
 - a. Length -
 - b. Operands -
 - c. Diagram -
 - c. Type Instructions -
2. Register to Storage (RS).
 - a. Length -
 - b. Operands -
 - c. Diagram -
 - d. Type Instructions -
3. Register to Indexed Storage (RX).
 - a. Length -
 - b. Operands -

- c. Diagram
- d. Type instructions -
- 4. Storage and Immediate (SI).
 - a. Length -
 - b. Operands -
 - c. Diagram -
 - d. Type Instructions -
- 5. Storage to Storage (SS)-
 - a. Length -
 - b. Operands -
 - c. Diagrams -

d. Type Instructions -

VI. Interrupts - The mechanism that allows the machine to change from the problem program state to the supervisor state.

A. Types of Interrupts.

- 1.
- 2.
- 3.
- 4.
- 5.

680

B. Priority of Interrupts for Simultaneous Occurances.

1. Stacking Priority.

- a.
- b.
- c.
- d.

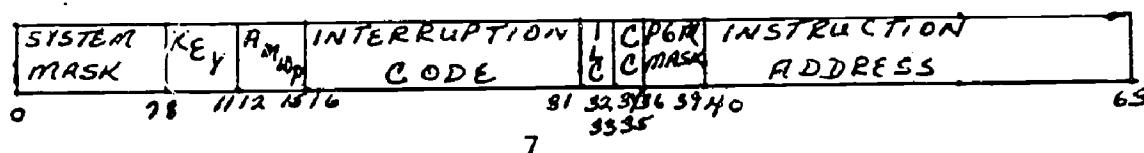
2. Execution Priority.

- a.
- b.
- c.
- d.

3. Handling of Simultaneous Interrupts

C. Program Status Word (PSW).

- 1.
- 2. Current PSW -
- 3. New PSW -
- 4. Old PSW -
- 5. Diagram -



IA-01-01-05

D. Interrupt Processing.

1.

2.

3.

4.

5

VII. Sys/360 Input/Output

A. IOCS. IOCS stands for InterOutput Control System. In the supervisor there is a small block of coding which serves to initiate I/O operations in response to program requests. It also notifies your program that an I/O operation is complete.

B. PIOCS. Physical IOCS is the most basic method of accessing the IOCS section of the supervisor. In PIOCS the programmer has direct control over I/O operations but is responsible for all error checking and program notification of completed I/O.

C. LIOCS. Logical IOCS is the system of accessing I/O devices most often used. LIOCS provides for all error handling and program notification of completed I/O. In order to do this there are 3 parts to the LIOCS system:

1. A table used to define the appearance of the file being used. This table is called a Define The File table or DTF.
2. A block of executable coding to handle error checking, I/O initiation, etc., called a module.
3. An I/O imperative specifying reading or writing on the file.

Keywords: TABULAR-MODULAR STRUCTURE, I/O AREA, DEVICE ADDRESS, MODULE NAME, PROGRAMER OPTIONS

In order for the 360 to perform an I/O operation three things must be done:

1. Specific information relating to the file must be made available in a table called a DTF table.
2. Coding (a piece of program) must be available to reference the table and pass the information to the channel. This block of coding is called an I/O module.
3. A specific command must be given to initiate the process.

The following information is needed in the table:

1. A Device Address consisting of a logical unit address must be provided. This address is then used to locate the physical address of the unit.
2. The location of the data must be specified explicitly. This area is called an I/O Area and is first defined by the programmer. This area is used to hold data for output or if it is an input area to receive data. Usually there is at least one I/O area per file.
3. Any special device dependent options are also specified. These are called programer options. An example would be how a tape is blocked or whether a tape is to be rewound at the end of the job.
4. The assembler takes into account all of the above information and determines which relocatable module will do the job.

I/O Module

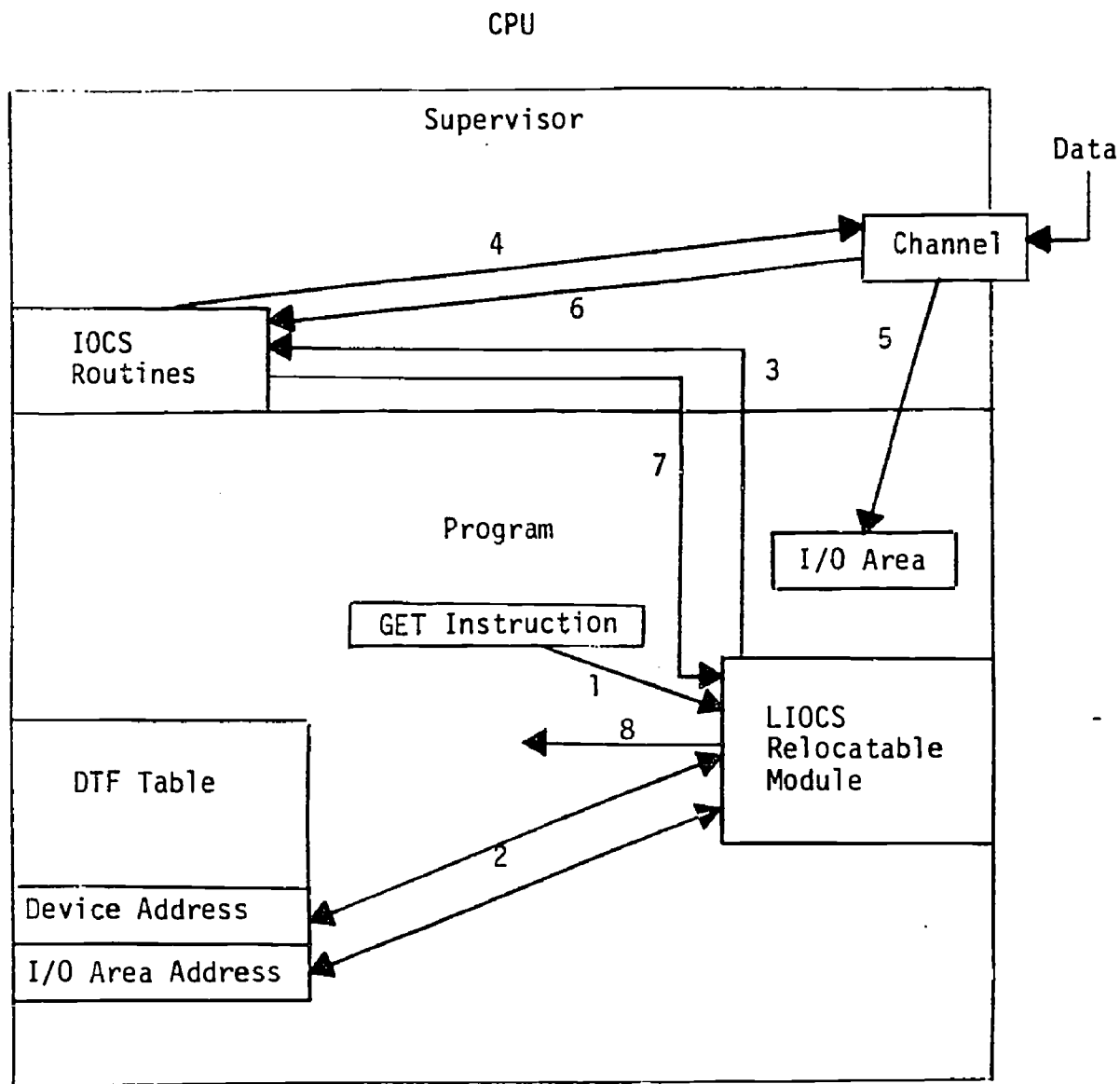
The I/O module is resident in the relocatable library. There are I/O modules for each set of options for each device.

The linkage editor is given the name of the appropriate module in the external address portion of the symbol table. It then pulls it off the relocatable library and links it in to your program.

The I/O module provides an interface between your program and the IOCS routines in the supervisor. It also provides for error checking and program notification of completed I/O. It also provides the coding necessary for implementation of programer options.

Imperative Command

In order to access the table and module for I/O the programmer need only use the GET (filename) or PUT (filename) commands. These commands provide branches to the appropriate I/O routines.



1. GET instruction branches to module.
2. Module obtains information it needs from DTF table.
3. Module passes information to IOCS routines in supervisor and requests I/O initiation.
4. IOCS routine passes information to channel and starts I/O.
5. Channel places data in I/O area in core.
6. Channel notifies IOCS of I/O completion.
7. IOCS passes completion information to module. The module then analyzes completion code to determine what error conditions, if any, are present.
8. If no errors, control is passed back to program.

Computer Systems Command Interface
With the Field Data Processing Activity

A Self-Paced Referenced Text

Programmer/Analyst Course
Software Division, Computer Science Department

January, 1979

IA-01-04-26

INTRODUCTION

OBJECTIVE: Given a situation at a field DPA running standard Army systems, be able to report an incident or a recommended system change on the proper forms and specifying the required information.

CRITERION TEST: The test for this class is different from the others in the course in several ways. First of all, the test is on a go/no go basis, that is, you either get all the points or you get none of them. Secondly, the test is designed to have you demonstrate your ability to perform the two primary CSC interface tasks. You will be given a copy of the test at the beginning (not the end) of the class time allotted to this block. Finally, the test will be graded with you present and, if it is not satisfactory, you will be told what is incorrect and asked to do the test over. This is because we believe that a test should be a learning tool as well as a demonstration of your performance.

METHOD OF LEARNING: This is a self-paced module and as such, you determine how fast you want to learn. If you have any questions on this topic, you may talk with other students or with the instructor. When you have finished the body of this text, complete the criterion test and take it to the instructor for one-to-one feedback. The only restrictions are the test must be finished by the end of the class time allotted and you may not talk with other students when taking the exam.

IA-01-04-26

PURPOSE

This text was designed to acquaint you with how the Army maintains it's standard software systems and the part that you, the programmer, play in the overall scheme. Speaking in more specific terms, this class will show you how to report problems with systems that were not written by you but are running at your DPA. By the way, if you do not know what that abbreviation means, there is a glossary in the back of this text that was designed for people who are not familiar with the Army's unique ADP terminology. Chances are that if you find other unfamiliar terms during this class you will find them defined in the glossary also.

WHY SOFTWARE STANDARDIZATION?

Although the Army has many different units scattered throughout the world, their data processing requirements are really quite similar. They all have, for instance, the following needs:

- * Keep personnel data current and available.
- * Requisition supplies.
- * Keep soldier's pay data current.
- * Maintain accountability of government property.

If each Army DPA had to develop software systems to accomplish these and other tasks, there would have to be dozens of programmers at the DPA and lots of spare run time to compile and test programs. Additionally, people moving from Post to Post (a fact of Army life) would have to spend many hours being retrained on new systems that did basically the same tasks as the systems at their last Post. With a lack of standardization among ADP systems, attempting to send data between these systems would be a nightmare.

CSC COMES INTO THE PICTURE

To alleviate these problems, the Army designated the Computer Systems Command (CSC) as the central design agency for all ADP systems that are run at more than one command. Under direction of the functional proponent (it's in the glossary), CSC develops standard software systems and sends them to all DPA who have a requirement for them. Not all DPA run the same standard systems, (for instance: a DPA at an Infantry Division would not have the same ADP requirements as the DPA that supports Fort Harrison). When CSC sends a system out it comes in a package containing operator's documentation, JCL, executable phases and instructions on how to load the phases on the libraries. Notice that there are no source programs or programmer documentation in the package. This is to preclude individual commands from changing the programs and therefore not having the same system as other commands. Some of the systems developed by CSC that you are likely to encounter are:

IA-01-04-26

3
687

- * SIDPERS - Maintains complete personnel data and prints personnel reports. Used at all Divisions and Posts.
- * SAILS - Standard supply system at Posts, Corp and supply inventory and management centers.
- * STANFINS - The system that keeps track of Army spending at each Post.
- * DLOGS - Used at all Divisions for requesting supplies and maintaining property records.

These are just some of the systems that CSC is responsible for. They actually have many systems varying:

- * in complexity from under 10 programs to over 100.
- * in Hardware from a UNIVAC 1005 to an IBM 370.
- * in scope from commissary management to Army Force Accounting.

WHAT DO I DO?

At this point you may be asking yourself what could you possibly have to do with all this since all CSC systems have already been developed and made operational. The answer to this question is why you are having this class!

When you go to a permanent assignment at a DPA you will probably be responsible for assuring the correct operation of an AMIS (remember the glossary!). Your duties in this area will be to:

- * Load system changes to your libraries to keep the system current.
- * Troubleshoot problems occurring during the execution of an AMIS, and report problems that cannot be overcome to the Assigned Responsible Agency (ARA).
- * Forward recommended changes of an AMIS to the ARA.

IA-01-04-26

We will cover each of these tasks separately in the following paragraphs.

KEEPING THE SYSTEM CURRENT

Just as the Army requirements and standards change from time to time so do the AMIS that support the Army. CSC produces system change packages (SCP) on a regular basis, usually every 3 - 6 months, that will keep the system current with Army policy. An SCP comes with all necessary documentation and load instructions. All that is required of the programmer (you) is to forward the package to the operators with any local run instructions and review the output from the SCP load to make sure that the system was loaded correctly. Following the load, you must notify CSC of the date and time of the load. You should be sure to load the SCP as soon as possible so that your system is the current one.

Occasionally, there will be emergency requirements to change a system, usually due to a program failure. When this occurs, CSC sends an Emergency/Urgent Change Package (EUCP) to all affected DPA. EUCP are normally not mailed but are sent through the Automatic Digital Network (AUTODIN) and will arrive within 2 days of dispatch. These change packages must be loaded immediately upon receipt.

WHAT IF THE SYSTEM DOESN'T WORK RIGHT

If an AMIS does not work right, either from an ADP standpoint or a functional standpoint, this means that an incident has occurred. Analyze the incident to make sure it is not caused by a local problem. Once you determine that the problem is not local, report it to CSC as soon as possible by phoning the Customer Assistance Office (CAO). The information that the CAO needs can be taken directly from the Incident Report form, (USACSC Form 53) that you must fill out for your records. Let's take a look at this form in detail, line by line. Refer to the sample form while you read the items below.

1. a. Enter your post name.
- b. This is a number consisting of three fields:
 - The AMIS code
 - Your DPA code
 - A three position sequence numbera sample number would be: C02-R111-027

IA-01-04-26

c & d. This code designates your DPA type and will be furnished you at your DPA.

e & f. Your name and phone.

g. The hours (in local time) when you work.

h. Your name (unless someone else phones the incident to CSC).

2. a. Leave blank.

b. AMIS Name.

c & d. Self-evident.

e. The person you talked to at CSC.

f. CAO phone number.

3. a. If the incident caused the job to blow-up indicate that and enter the termination code printed on the output listing.

b. Self-evident.

4. This is the number of the SCP or EUCP that was loaded most recently and the date it was loaded (installed). If the problem involves a program, enter the program name in block A or if the problem is in the system documentation enter the documentation name in block B.

5. a. Enter the job name here.

b. Explain in detail exactly what happened. This is very important because it is from this description that CSC will start to resolve the problem.

c. Explain what you did, if anything, to get around the problem.

d. What effect does this incident have on you?

e. Leave blank.

6. Most often this will be applications software.

7. Which of these items are available for CSC to refer to, if needed.

You should maintain a file of all incident reports and indicate in that file the action taken by CSC to resolve the problem.

INCIDENT REPORT

603

For use of this form, see USACSC REG 18-21-1; the proponent agency is Quality Assurance Directorate, (Test & Conf Mgt Div)

1A. INSTALLATION <u>Fort Harrison - USAIA</u> B. ORIGINATOR NUMBER <u>P02-R111-006</u> C. ORIGINATOR LEVEL: <u>1</u> D. OPERATING ENVIRONMENT: <u>011</u> E. CONTACT <u>SP4 Smith</u> F. CONTACT'S PHONE <u>699-2323</u> G. CONTACT WILL BE AVAILABLE <u>0730-1630</u> H. PERSON PHONING IN INCIDENT <u>SP4 Smith</u> PHONE _____	2A. CAO ID _____ B. FUNCTIONAL AMIS <u>SIDPERS</u> C. DATE RECEIVED <u>12 Jan 79</u> D. TIME RECEIVED <u>1600</u> E. RECEIVED BY <u>SFC Jones</u> F. PHONE <u>354-1616</u>
3. OPERATING STATUS AT TIME OF INCIDENT A. WAS THERE A CYCLE HALT OR ABNORMAL TERMINATION IN THE PRODUCTION RUN: <input checked="" type="checkbox"/> YES ABEND CODE <u>WRONG LENGTH RECORD</u> <input type="checkbox"/> NO B. DATE/TIME INCIDENT DETECTED: <u>1200 Jan 79, 1500</u>	4. BASELINE <u>P02-21-05</u> INSTL'D <u>5 Jan 79</u> A. PROB PGM ID <u>P3AAAC</u> VERS _____ OR B. PROB DOC ID _____ CHG LEVEL _____ PARA(S) _____ PAGES _____
5. DESCRIPTION OF INCIDENT. A. ID OF RUN IN PROGRESS <u>DAILY CYCLE</u> B. NARRATIVE DESCRIPTION OF INCIDENT AND RELATED EVENTS <u>Job aborted due to wrong length record on input tape A3AAAC. Listed tape and saw that all records were one byte too long.</u> C. ACTION TAKEN BY USER AND RESULTS <u>Used TPTP utility to strip the extra byte and continued.</u> D. IMPACT ON USER UNTIL RESOLVED <u>Every time the cycle is run, the tape must be reformatted.</u> E. CAO/ASD REDEFINITION OF PROBLEM IF DIFFERENT THAN 5B. 	

6. TYPE OF DIFFICULTY		7. SUPPORTING DOCUMENTATION FOR RESEARCH BY USER			
X	APPLICATION SOFTWARE	AVAIL		AVAIL	
	EXECUTIVE SOFTWARE	X	CORE DUMP		OUTPUT FILE TAPE
	INPUT DATA	X	CONSOLE COPY		DATA FILE TAPE
	DOCUMENTATION		JOB CONTROLLIST		RPT OUTPUT TAPE
	HARDWARE	X	SYSLST OUTPUT	X	DOCUMENTATION
	OTHER	X	INPUT FILE TAPE		OTHER _____
			DISK MODULE		

I KNOW A BETTER WAY

After you have had an opportunity to observe an AMIS in it's normal operation, you may think of ways to improve it. Additionally, the people who use the output, the functional user, may like to see the output changed or possibly new reports produced. CSC encourages these suggestions and the Army has provided a means of submitting them: the System Change Request (SCR). There are two types of change requests, technical, involving a suggestion for technical ADP reasons and functional, involving a suggestion to change the input or output from the customer's viewpoint. You will be responsible for filling out technical SCR and you should review functional SCR after they have been prepared by the user. You should keep a log of all SCR you have submitted. CSC distributes a monthly SCR status report which you can compare to your SCR log to make sure that your information agrees with CSC's information.

Refer to the sample SCR and we will discuss the blocks you will have to fill out.

Block 1. This will be the address of the functional proponent of the system.

Block 2. The address of your DPA.

Block 3. This number has the same format as the originator number in the incident report.

Block 4. Your name.

Block 5. You can only check routine. The other three blocks are for the functional proponent and CSC.

Block 6. The AMIS name and, if applicable, the program name.

Block 7. If this SCR is the result of an incident, then complete this block. Otherwise leave blank.

Block 8. Self-evident.

Block 9. If the SCR involves system documentation, check the appropriate block.

Block 10. Attach any documents that may help to explain the suggestion and check the appropriate block.

IA-01-04-26

Block 11. Enter the problem as it currently exists in part A. In part B describe the solution you are suggesting in sufficient detail so that it can be clearly understood.

Block 12. Usually you will have to furnish a copy of the SCR to CSC and to your major command MISO.

Block 13. Your name, job and signature.

Block 14. Leave blank.

IA-01-04-26

70

693

606

1. TO: HQDA (DACA-CSS-F) Pentagon Washington, DC 20310	2. FROM: MISO Fort Harrison, IN 46216	3. ORIGINATOR NO: F01-R111-016			
5. CATEGORY (CHECK ONE): <input type="checkbox"/> EMERGENCY <input checked="" type="checkbox"/> ROUTINE <input type="checkbox"/> URGENT <input type="checkbox"/> PRIORITY		4. POINT OF CONTACT: SP5 Johnson 7. INCIDENT ENCOUNTERED STATION _____ DATE _____ TIME _____			
6. SI/BSYSTEM <u>VTAAAS</u> PROGRAM ID <u>P05ANV</u> VERSION NO _____					
8. SHORT TITLE: (30 CHARACTERS MAXIMUM INCLUDING SPACES). Operator MSG in Program P05ANV					
9. DOCUMENTATION IDENTIFICATION					
A. DPI USER MANUALS <input type="checkbox"/> B. FUNCTIONAL USER MANUALS <input type="checkbox"/>	C. EXECUTIVE SOFTWARE <input type="checkbox"/> D. FUNCTIONAL SOFTWARE <input type="checkbox"/>				
10. ATTACHMENTS					
A. MAPS <input type="checkbox"/> B. CORE DUMPS <input type="checkbox"/> C. IMPACT STATEMENT <input type="checkbox"/>	D. FILE PRINTOUTS <input type="checkbox"/> E. CONSOLE SHEETS <input checked="" type="checkbox"/> F. DFSR <input type="checkbox"/>	G. OUTPUT LISTS <input type="checkbox"/> H. JOB STREAM SEQ. <input type="checkbox"/> I. OTHER <input type="checkbox"/>			
11. NARRATIVE: A. PROBLEM DESCRIPTION: Program P05ANV prints 5 lines of operator messages. This takes excessive console and operator time. B. RECOMMENDED SOLUTION/ACTION TAKEN: Delete the unnecessary words and abbreviate the remainder so that only 1 or 2 lines are printed.					
12. COPY FURNISHED: HQ TRADOC MISO USACSC; CSCS-QAC-M	13. PREPARED BY: SP5 Daniel Johnson Programmer/Analyst				
DATE: _____	SGNR: _____ DATE: _____				
14. PROPONENT AGENT REVIEW: <table style="width: 100%;"> <tr> <td style="width: 33%;"> A. TYPE OF CHANGE <input type="checkbox"/> FUNCTIONAL <input type="checkbox"/> TECHNICAL </td> <td style="width: 33%;"> B. CLASS OF CHANGE <input type="checkbox"/> REGULATORY <input type="checkbox"/> NON-REGULATORY </td> <td style="width: 33%;"> C. EXTENT OF CHANGE <input type="checkbox"/> MAJOR <input type="checkbox"/> MINOR </td> </tr> </table> D. REFERRED TO ARA FOR ANALYSIS (DATE): _____ E. DISPOSITION: <input type="checkbox"/> APPROVED, REQUESTED IMPLEMENTATION: _____ <input type="checkbox"/> DISAPPROVED F. FUNCTIONAL GUIDANCE: <input type="checkbox"/> ATTACHED <input type="checkbox"/> NOT REQUIRED <input type="checkbox"/> TO BE PROVIDED BY _____			A. TYPE OF CHANGE <input type="checkbox"/> FUNCTIONAL <input type="checkbox"/> TECHNICAL	B. CLASS OF CHANGE <input type="checkbox"/> REGULATORY <input type="checkbox"/> NON-REGULATORY	C. EXTENT OF CHANGE <input type="checkbox"/> MAJOR <input type="checkbox"/> MINOR
A. TYPE OF CHANGE <input type="checkbox"/> FUNCTIONAL <input type="checkbox"/> TECHNICAL	B. CLASS OF CHANGE <input type="checkbox"/> REGULATORY <input type="checkbox"/> NON-REGULATORY	C. EXTENT OF CHANGE <input type="checkbox"/> MAJOR <input type="checkbox"/> MINOR			
SIGNED: _____ DATE: _____					

607

WHAT NOW?

We hope that you now have an understanding of your relationship in a DPA to the Computer Systems Command. Incident and recommended change reporting will be a fact of Army life to you.

IA-01-04-26

12

695

GLOSSARY

1. Army Management Information System (AMIS). This term refers to any software system used in the Army. When used in this text, however, the term refers to standardized software that is designed, tested and programmed at Computer Systems Command and run at many DPA. Sometimes this type of AMIS is called SAMIS (Standard AMIS) or Class A AMIS.
2. Assigned Responsible Agency (ARA). The agency who designs, programs and tests AMIS. CSC is normally the ARA for Class A AMIS.
3. Automatic Digital Network (AUTODIN). A communications link among most Department of Defense activities used for transmitting data from post to post. Data is transmitted electronically (similar to a phone call) and is received either in punched cards or magnetic tape.
4. Computer Systems Command (CSC or USACSC). This is the Army's software design and planning agency. It develops, programs and tests centralized Class A AMIS and plans for the future direction of Army software.
5. Customer Assistance Office (CAO). This is the central contact point that CSC has established for reporting incidents. It is generally manned 24 hours a day. The CAO will take all the information on the incident report and forward it to the responsible programmer for resolution.
6. Data Processing Activity (DPA). Any Army organization which operates data processing equipment. This can range from a unit with just data entry equipment to a unit with several large scale computer systems.
7. Emergency/Urgent Change Package (EUCP). Changes to standard AMIS sent by CSC to resolve operational difficulties. This is the way CSC distributes changes made as a result of incident reports.
8. Functional Proponent. The organization which is responsible for the functions of an AMIS. For example the Deputy Chief of Staff for Personnel is the functional proponent for SIDPERS. Normally HQDA staff offices are the functional proponents for CSC systems.
9. Functional User. The activity which uses the output from an AMIS; usually non-ADP personnel who are customers of the DPA.
10. Incident Report (IR). Reporting a system operational problem, that is, where the system does not work as it should, to CSC via telephone. This is also the written record (USACSC form 53) made in support of the phone call.

11. System Change Package (SCP). A regularly scheduled change to a standard AMIS developed by the ARA and sent to all DPA who run the affected system. An SCP will normally include changes made as the result of SCR's.
12. System Change Request (SCR). A request to either the PA or ARA of a standard system to make a change in the system. If the change is for ADP efficiency, it is called a technical SCR; if it is to change data on initial input or output it is called a functional SCR.

HARDWARE/SOFTWARE SYSTEMS EFFICIENCY

SCENARIO

1. SITUATION:

a. As previously stated, you have been assigned to this organization as the Systems Software Analyst, responsible for the operating system and associated software functions.

b. New equipment configuration.

- | | | |
|------|---------------------------|------|
| (1) | IBM 360 Model 30 - 64K | |
| (2) | 1050 Console Keyboard | 1 ea |
| (3) | 2314 Disk w/9 drives | 1 ea |
| (4) | 2400 Tape Drive (9 track) | 4 ea |
| (5) | 2804 Control Unit | 1 ea |
| (6) | 1403 Printer | 1 ea |
| (7) | 2540 Card Read/Punch Unit | 1 ea |
| (8) | 2821 Control Unit | 1 ea |
| (9) | Selector Channel | 2 ea |
| (10) | Multiplexer Channel | 1 ea |

c. Special Features

- (1) Floating Point
- (2) Decimal
- (3) Timer

d. To support

- (1) Multiprogramming
- (2) COBOL
- (3) FORTRAN
- (4) PL/I (48 character set)

e. Manufacture Supplied Packages

- (1) Operating System Routines (Supervisor, Linkage Editor, etc)
- (2) Subroutines for FORTRAN and PL/I
- (3) Utility Programs

f. User systems now in effect are outlined in the separate systems charts, incl 1 and 2. Memory requirements and run times for each process step are indicated on the chart. Additional programs to be supported are listed in incl 3 and are to be assumed to be stand-alone.

2. REQUIREMENTS

a. Design and generate, on paper, an operating system specifying selected options, core requirements, and total size of your supervisor. Calculate the "Send"Address.

	OPTIONS/SELECTIONS	MAIN STORAGE REQUIREMENTS
SUPV		
SYSTEM=	_____	_____
MPS=	_____	_____
TP=	_____	_____
MICR=	_____	_____
ASCII=	_____	_____
AP=	_____	_____
EU=	_____	_____
ERRLOG=	_____	_____
MCRR=	_____	_____
CONFIG		
MODEL=	_____	_____
SP=	_____	_____
DEC=	_____	_____
FP=	_____	_____
TIMER=	_____	_____
PORT=	_____	_____
STDJC		
DECK=	_____	_____
LIST=	_____	_____
LISTX=	_____	_____

Options/Selections

Main Storage Requirements

SYM=

XREF=

ERRS=

CHARSET=

LOG=

DUMP=

LINES=

DATE=

SPARM=

FOPT

OC=

IT=

PC=

TEB=

TEBV=

EVA=

SKSEP=

CE=

JA=

JALIOCS=

PTO=

IDRA=

OLTEP=

RETAIN=

PCIL=

CBF=

Options/Selections

Main Storage Requirements

CCHAIN=	_____	_____
TRKHLD=	_____	_____
AB=	_____	_____
WAITM=	_____	_____
DASDFP=	_____	_____
SYSFIL=	_____	_____
PIOCS	_____	_____
SELCH=	_____	_____
BMPX=	_____	_____
CHANSW=	_____	_____
TAPE=	_____	_____
MRSLCH=	_____	_____
IOTAB	_____	_____
BGPGR=	_____	_____
F1PGR=	_____	_____
F2PGR=	_____	_____
JIB=	_____	_____
CHANQ=	_____	_____
IODEV=	_____	_____
TOTAL		_____

Send Address: SEND _____

615

b. List available channels and distribution of I/O devices:

<u>Type Channel</u>	<u>Channel Number Assigned</u>	<u>Devices</u>
---------------------	--------------------------------	----------------

c. Specify size of required partitions:

ALLOC F1= _____ K, F2= _____ K

d. Prepare a schedule of jobs for a Tuesday in which the payroll checks are to be run. Shift starts at 0700 hrs and we run two shifts of eight hours each. Indicate Program Number, Time to be Run and Partition. Schedule should be such that the first job on the schedule is the first job to be run, etc.

Program Number

Time to be Run

Partition

617

e. List contents of each library in specific terms, by routine name or program number.

(1) Core Image Library

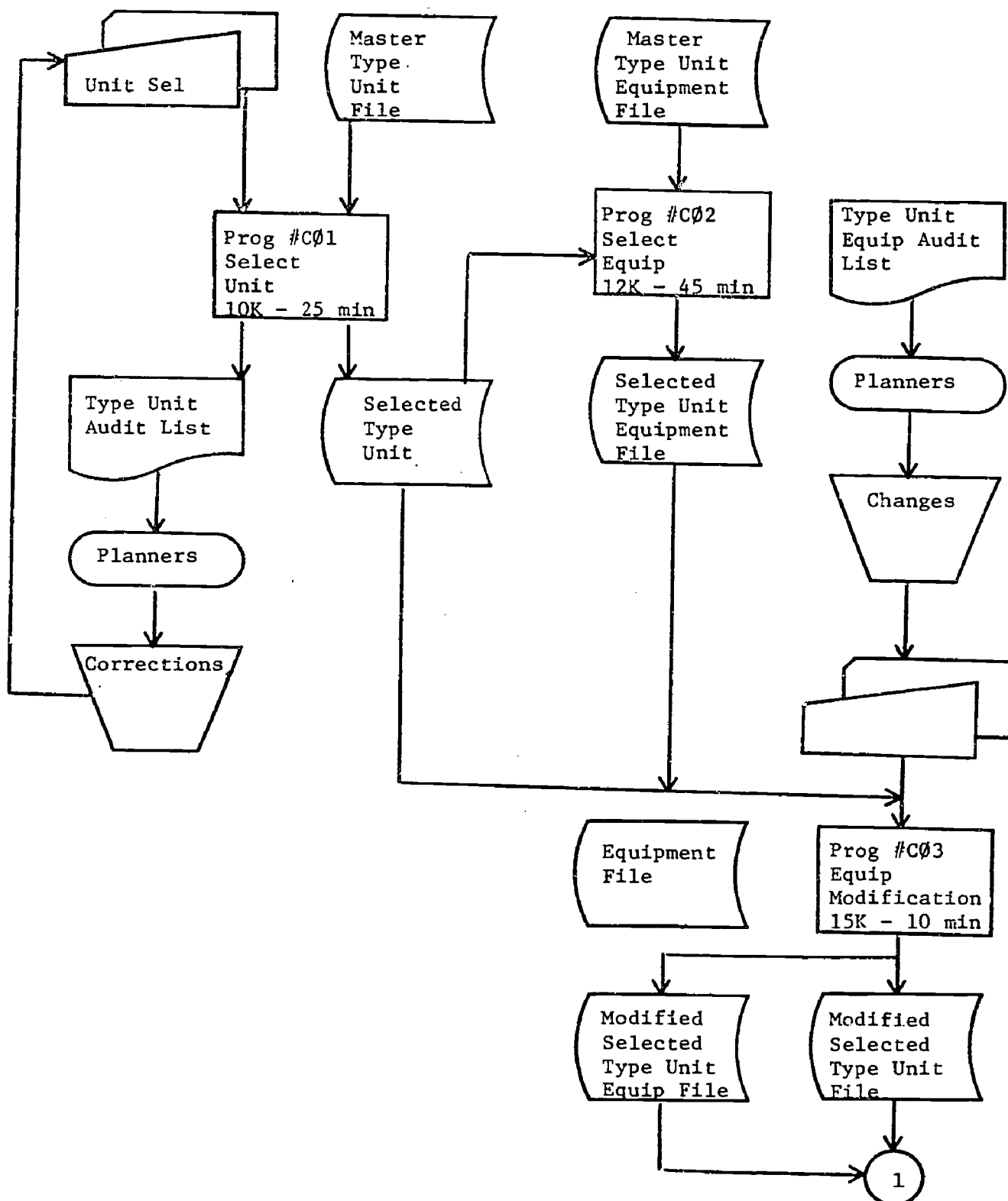
(2) Relocatable Library

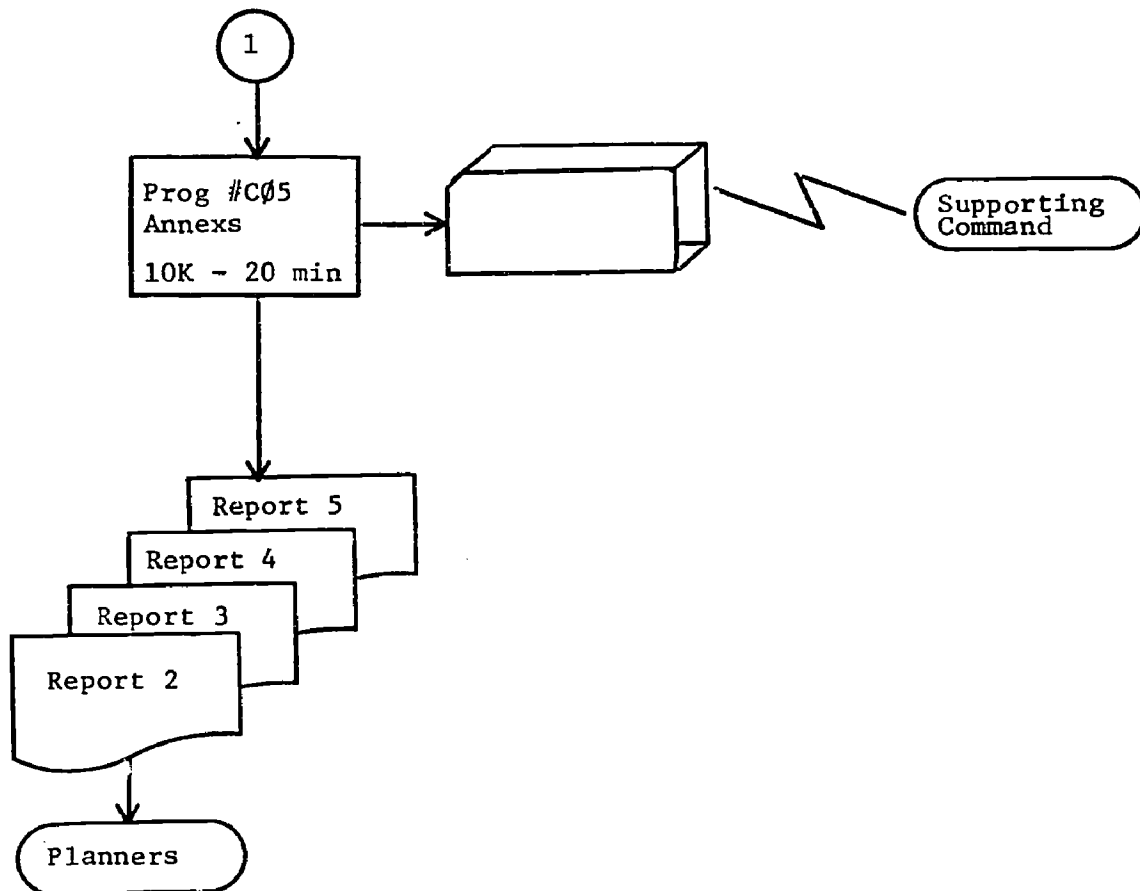
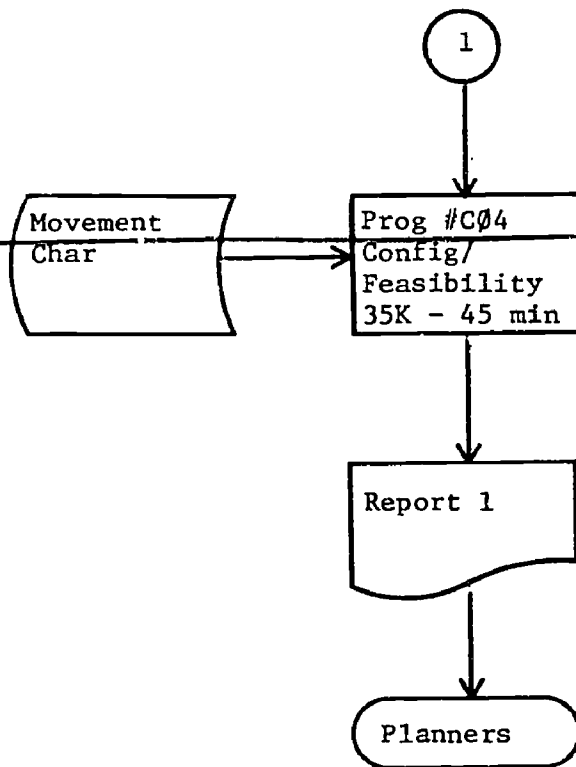
(3) Source Statement Library

f. Does your system require "Spooling"? Yes _____ No _____ Why? _____

CONTINGENCY PLANNING

FREQ: As Required



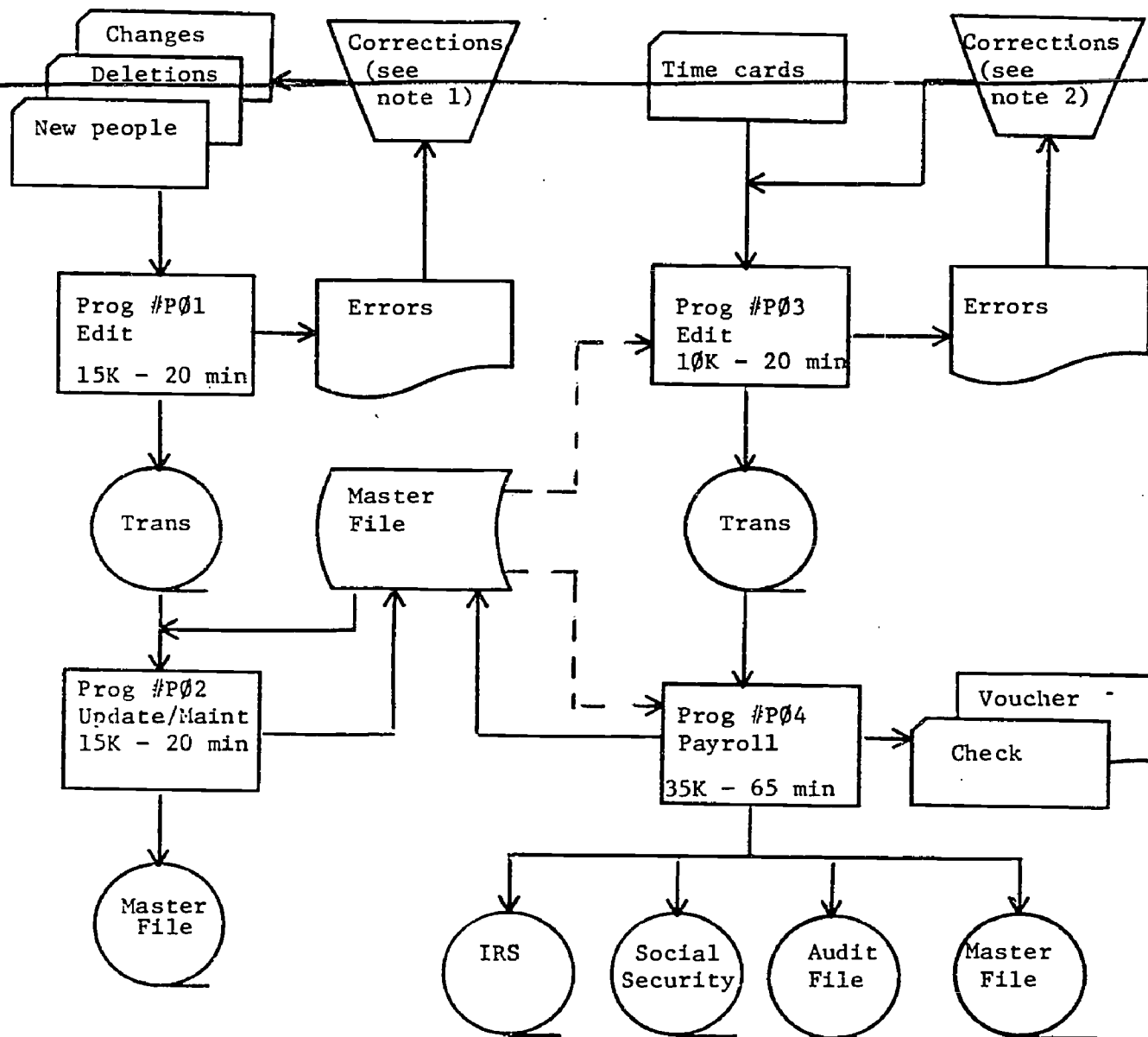


Incl 1 - pg 2

CIVILIAN PAYROLL

DAILY

BI-WEEKLY



Note 1: The errors are to be corrected and the corrections inputted into the next day's cycle, except those errors of the last cycle for the period are to be corrected immediately and the edit rerun. All errors must be corrected before the Payroll can be run.

Note 2: The errors must be corrected immediately and the edit rerun until no errors exist. Only then will the payroll checks be run.

Incl 2

ADDITIONAL PROGRAMS

Incl 3	PROG NR	SIZE K	FREQ	TIME REQD	RUN TIME	INPUT FILES			OUTPUT FILES			PRINTER
						TAPE	DISK	READER	TAPE	DISK	PUNCH	
	A01	27	Daily	1000 hrs	10	2	1	X		1		1
	A02	10	Daily	0800 hrs	3	1	1	X		1		1
	A03	33	Daily	2400 hrs	25	2	3	X	2	2		2
	A04	20	Daily	2400 hrs	57	1	2				2	1
	A05	15	Daily	1200 hrs	15		3	X				3
	A06	25	Daily	2400 hrs	30	2		X	1		1	2
	A07	35	Daily	2400 hrs	240	1	4	X		4	2	3
	A08	34	Tuesday	2400 hrs	65	2	4			2		3
13	A09	29	Friday	2400 hrs	30		2	X	1	1		1
	A10	15	Monday	2400 hrs	27	1	3			2	1	2
	A11	21	Wed	2400 hrs	32	2	4	X	2			4
	A12	17	Thur	2400 hrs	19		2		1		1	
	A97		Daily		120	Compile and Test Time						
	A98		Daily		60	Software Maintenance/Software Analyst						
	A99		Daily		60	Systems Maintenance/Manufactures Rep						
	Spooling	18	If determined to be required									

USAF PP 190-20 7/74

USAF PP 190-20 7/74